ABSTRACT VISUALIZATION OF LARGE-SCALE TIME-VARYING DATA

by

Li Yu

A dissertation submitted to the faculty of
The University of North Carolina at Charlotte
in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in
Computing and Information Systems

Charlotte

2012

Approved by:

____

Dr. Aidong Lu

____

Dr. William Ribarsky

____

Dr. Jianping Fan

____

Dr. Kalpathi Subramanian

____

Dr. Wei Chen

____

Dr. Yanqing Sun

ABSTRACT

LI YU. Abstract visualization of large-scale time-varying data. (Under the direction of DR. AIDONG LU)

The explosion of large-scale time-varying datasets has created critical challenges for scientists to study and digest. One core problem for visualization is to develop effective approaches that can be used to study various data features and temporal relationships among large-scale time-varying datasets.

In this dissertation, we first present two abstract visualization approaches to visualizing and analyzing time-varying datasets. The first approach visualizes time-varying datasets with succinct lines to represent temporal relationships of the datasets. A time line visualizes time steps as points and temporal sequence as a line. They are generated by sampling the distributions of virtual words across time to study temporal features. The key idea of time line is to encode various data properties with virtual words. We apply virtual words to characterize feature points and use their distribution statistics to measure temporal relationships. The second approach is ensemble visualization, which provides a highly abstract platform for visualizing an ensemble of datasets. Both approaches can be used for exploration, analysis, and demonstration purposes.

The second component of this dissertation is an animated visualization approach to study dramatic temporal changes. Animation has been widely used to show trends, dynamic features and transitions in scientific simulations, while animated visualization is new. We present an automatic animation generation approach that simulates the composition and transition of storytelling techniques and synthesizes animations to describe various event features. We also extend the concept of animated visualization to non-traditional time-varying datasets - network protocols - for visualizing key information in abstract sequences. We have evaluated the effectiveness of our animated visualization with a formal user study and demonstrated the advantages of animated visualization for studying time-varying datasets.

ACKNOWLEDGMENTS

This dissertation would not have happened without the help, inspiration, and encouragement of many people along the way.

First and foremost, I want to thank my advisors: Prof. Aidong Lu. I am very fortunate to have the privilege to be her graduate student. Over the last several years, she have always been ready for guidance not only in my research but also in life. Several of those late nights helping with my research paper will remain forever unforgettable. Her support went beyond the role as my advisors and I couldn't have asked for more.

I am honored to have Professors William Ribarsky, Jianping Fan, Kalpathi Subramanian, Wei Chen and Yanqing Sun serve on my PhD Committee. They provided helpful comments to improve this manuscript. In particular, I would like to thank Prof. Wei Chen for guiding me into the world of computer graphics and visualization and serve on my committee remotely from Zhejiang University. I am grateful to Prof. William Ribarsky, Kalpathi Subramanian and Jianping Fan for their ideas and help during my PhD study. Prof William Ribarsky and Kalpathi Subramanian helped me to build more background ideas on visualization and visual analytics and taught me how to do research and write research papers.

This research is supported by DHS Center of Excellence - Natural Disasters, Coastal Infrastructure and Emergency Management (DIEM), DOE DE-FG02-06ER25733, and NSF 0633150.

Last but not least, I would like to thank my family. My parents' encouragement and supports are the backbones of my constant struggle from frustration to empowerment.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1: INTRODUCTION

1.1 Motivation

Large-scale time-varying data visualization has become a necessary component of many science and engineering research fields, which are of great scientific interest as well as immense social-economic impact [128]. For example, models of storm surge and inundation can be used in applications ranging from hindcasts and forecasts of tidal circulation, forensic studies in the aftermath of hurricanes, to planning of new hurricane protection systems. Recently, visualizations of storm surge simulations are used to track nearshore oil spill movement in the Gulf of Mexico (adcirc.org/oilspill).

Scientific studies often require strong visualization tools to assist scientists to explore and analyze large-scale time-varying datasets. Current visualization techniques provide many effective approaches to studying a single dataset and it is acceptable to use a single dataset method to explore a small amount of datasets by visualizing each of them. However, the traditional methods are not efficient for investigating large-scale simulations. Even though some approaches allow users to take snapshots for later comparison or switch datasets during visualization, the limitation of the human perception system only allows us to visually compare and track several objects at the same time [138]. The increasing complexity of simulation datasets and various temporal relations also compound the challenges of Large-scale time-varying studies, which may otherwise contribute to new scientific discoveries.

The research work delivered in this dissertation is motivated by the fact that new visualization techniques are required to assist the analysis of such large-scale time-varying datasets. The fundamental problem is to design suitable visualization platforms for studying various data features and temporal relationships, ranging from photo-realistic 3D structures to abstract high-dimensional statistics.

1.2 Overview

This dissertation focuses on exploring large-scale time-varying datasets with abstract visualization approaches. The main purpose of abstract visualization is to represent various temporal relationships at different scales visually.

The main challenge comes from the variations of data features that range from static object shapes to temporal events, which are hard to describe or measure. Our approaches are generally built with feature extraction and event detection techniques, which are essential to interpret large-scale simulation datasets and reduce data sizes. Two main visualization platforms are designed for visual data exploration, comparison, and representation. With these visualization integrate data analysis method, scientist can gain more insights on the temporal relationship instead of focusing on individual time steps.

In this dissertation, we present two main abstract visualization frameworks as follows:

1. Time Line:

   The first framework is to generate succinct time lines to represent various temporal relationships at different scales. A time line visualizes time steps as points and temporal sequence as a line. They are generated by statistically sampling of the distributions of data features across time.

Chapter 4 describes the core of this approach. We use virtual words, in the form of high-dimensional vectors, to encode diverse data features of scientific simulation. The virtual words of time-varying datasets are generated through sample collections at feature locations and the statistics of virtual words are used to measure temporal relationships. We demonstrate the usages of time lines on several exploration and visualization tasks in real-life applications.

Chapter 5 presents an approach of ensemble visualization, which provides a highly abstract platform for visualizing an ensemble of time-varying datasets. A set of combined time lines are visualized and clusters are reduced with special care. The visualization of an ensemble provides a quick approach to identify interesting and abnormal events.

2. Animated Visualization:

   The second component of this dissertation is an animated visualization approach to study dramatic temporal changes. Animation has been widely used to show trends, dynamic features and transitions in scientific simulations, while animated visualization is new. Different from animation sequences that are generated by concatenating snapshots from individual time steps, an animated visualization allows users to visualize a time-varying dataset with an animation and adjust the focused regions, durations, and features interactively.

   Chapter 6 presents an automatic animation generation approach that simulates the composition and transition of storytelling techniques.

Chapter 7 describes our evaluation of the animated visualization approach with a formal user study. The results demonstrate the advantages of animated visualization over interactive visualization for studying time-varying datasets.

Chapter 8 extend the concept of animated visualization to non-traditional time-varying datasets - network protocols - for visualizing key information in abstract sequences. A digital Lego approach is described to use both static visualization and animated visualization to study network security protocols. The effects of our animated Lego visualization in studying and understanding security protocols.

1.3 Contribution

This dissertation explores several approaches for visualizing large-scale time-varying datasets. We focus on the topic of abstract and efficient representation of time-varying datasets. By taking advantage of feature extraction and statistics analysis techniques, important temporal relationships and features of a large-scale dataset can be revealed.

The main contributions of this dissertation are the following:

1. A time line visualization approach to generate meaningful lines representing temporal relationships of one or an ensemble of large-scale time-varying datasets.
2. An animated visualization approach to explore time-varying datasets with an algorithm to generate semantic animations as digital stories.
3. Evaluation of the presented abstract visualization approaches over traditional techniques.

CHAPTER 2: BACKGROUND

Time-varying data visualization [79] is an active research topic for the past decade. Researchers have been using volume rendering algorithms, hardware accelerating techniques to accelerate the visualization in the early year. We first introduce some research work in visualizing time-varying dataset from different aspects in the following section. Then we discuss about research area close to the methods used in this dissertation.

2.1 Time-varying Visualization

2.1.1 Multi-variate Analysis

Large-scale datasets are normally multi-variate and multi-dimensional, finding the correlation between different variables becomes an essential step to visualize and analyze the data. Jeffrey et al. utilized various temporal curves, clustering and segmentation to organize data, capture temporal behaviors and explore correlations in time-varying multivariate data [126]. Wang et al. measured information flow in the data and used time plot and circular graph to visualize the information transfer for an overview of information transfer relations among the variables [140]. Chen et al. used a sampling-based approach to create a static volume classification that summarizes the correlation in multivarite datasets [30]. Yuan et al. proposed a visual analytics framework to explore seismic event catalog data with satellite imagery data [160]. Blaas et al. presented an interactive tool to enhance the usability of Parallel Coordinates with linked views and GPU acceleration for the exploration of large, multi-timepoint volumetric data sets and also preserving spatial context of the volumetric data [20]. Glatter and Huang developed a textual pattern matching approach for specifying, identifying general temporal patterns and allowing uncertainty to be used in the specification of patterns of interest, particularly ones that are temporal and multivariate in nature [63, 64]. Walker et al. presented a novel coupling of parallel coordinates with spherical coordinates to visualize the vector and multi-dimensional data. They enhanced visual perception, and represented vector data in a more natural spatial domain [136]. Aigner et al. proposed a technique called STZ that is capable of displaying quantitative data and qualitative abstractions of time-oriented, multivariate data. It used a combined representation of different visual encodings, whereas spatial position is used to encode the quantitative data and color-coding is used to display the related qualitative abstractions. They also evaluated their work by performing a user study which shows the composite representation were faster, particularly for more complex tasks [7].

2.1.2 Rendering

Better rendering of time-varying datasets will help enhance the perception as well as rendering the evolution of the data. Jankun-Kelly and Ma generated static transfer functions for time-varying data by merging several transfer functions over time [77]. Woodring et al. simulated the chronophotography technique to depict time-varying data features using a high dimensional direct rendering method [153]. Hsu et al.

depicted temporal behaviors using image-based methods and use illustrative visualization techniques to effectively present the evolution of 3D flow in a single 2D image [73]. Shi et al. proposed a multivariate visual analytics system by computing local and global properties of path lines of a 4D dataset describing relevant features of them and applying several information visualization techniques [122]. Akiba et al. proposed an approach that allows simultaneous classification of the entire time series and discuss interactive classification by exploring options for simultaneous rendering of the time series based on the time histogram [9].

2.1.3 Interactive Analysis

Interactive visual analysis also becomes important during the exploration and visualization of the dataset. A good interactive technique will help scientists to observe interesting patterns and events faster and easier. Kehrer et al. presented a system to do interactive visual analysis of two heterogeneous parts of scientific data [85]. Keefe et al. presented an interactive framework for exploring spacetime and form-function relationships in experimentally collected high-resolution biomechanical data sets [83]. Akiba and Ma developed an multi-view interface visualizing time-histogram, correlation between variables and spatial domain rendering to efficiently visualize large-scale dataset and support interactive analysis and exploration [10]. Sanyal et al. described a tool named Noodles which uses coordinated view of ribbon and glyph-based uncertainty visualization to visualize ensemble uncertainty [115].

2.1.4 Visual Design

Better visual designs could help user to gain better insights of the data. Bruckner and Moller uses sampling and spatio-temporal clustering techniques to generate a concise overview for the visual exploration of parameter spaces [27]. Balabanian et al. proposed temporal compositors that created a temporal characteristic for the spatially overlapping voxels to condense several time-steps into a single image [16]. Hoshi and Rheingans evaluated the effectiveness of their illustrative techniques by conducting a user study. They compared the ability of subjects to visually track features using snapshots, snapshots augmented by illustration techniques, animations, and animations augmented by illustration techniques. The results indicated that the use of illustration-inspired techniques provides a significant improvement in user accuracy and the time required to complete the task [82]. Fischer et al. designed a visualization technique called ClockMap for effective visualization of hierarchical time-series data. It combined a circular nested treemap layout with a circular glyph representation for time-series data [54]. Jang et al. used the functional representation approach for time-varying data sets and developed an efficient encoding technique utilizing temporal similarity between time steps [75].

2.1.5 Acceleration

Chiueh and Ma proposed a parallel pipelined renderer for time-varying volume data in 1997 [31]. Shen used a temporal hierarchical index tree structure [119] and Time Space Partitioning Tree [120] to store and render volume dataset. But due to the increasing size and dimension of time-varying datasets, processing, rendering time and interactivity create challenges during the analysis and visualization of the data. These techniques became impractical. Several other techniques have been proposed for visualization and comparison purposes [126, 6, 81].

Tikhonova et al. used RAF(Ray Attenuation Functions) to generate compact representation of volume data and explored transfer function space in a coherent manner without accessing the original 3D data [129]. Du et al. proposed a structure called space-partitioning-time (SPT) tree to achieve efficient volume rendering with higher re-use rate and more details [46]. Shen et al. developed a TSP tree to capture spatial and temporal coherence of the time-varying data for fast rendering [121]. Wang and Chiang presented a method using Persistent Time-Octree (PTOT) indexing structure for isosurface extraction and view-dependent filtering from large time-varying fields [139]. Wan et al. presented a new metric to measure the importance of volume datasets and automatically adjust the number of samples to produce temporally coherent sampling patterns [137]. Yu et al. pointed out that in-situ visualization is a promising direction for accelerating high-performance supercomputing and scientific discovery by performing a case study of in situ data processing and visualization strategies in a massively parallel environment [156]. Wang et al. proposed a method to compress a given large time-varying data set using an octree. The compression started with spatial partitioning and temporal grouping, which took into account domain knowledge that scientists provide. This solution significantly reduced the amount of data transferred and efficiently used the limited graphics memory [142].

2.2 Feature Extraction

Feature extraction methods are applied to the original time-varying datasets to accelerate analysis process. We use volumetric SIFT algorithm to select feature points for 3D datasets. The SIFT algorithm was originally proposed in image processing community to detect and describe local features in images. It has been applied successfully to different applications, such as object recognition [92], point tracking [88], panorama creation [26], medical imaging [102], and knowledge-assisted visualization [101]. Some approaches of feature tracking are also related to this topic, since they can provide the frame-to-frame correspondence between objects-of-interest to reveal the temporal trend of a time-varying dataset. The tracking information can be further studied to detect significant data changes. Currently, most feature tracking approaches are based on pre-defined feature models or user-specified regions-of-interest. The matching of data features is generally achieved by the following two mechanisms. First, based on selected regions-of-interest for feature tracking, either data features are matched based on their corresponding positions [123] or topological features are tracked using high dimensional geometries [78]. Critical points of geometry models have also been studied in many applications [62, 124, 47, 57]. Second, feature attributes, such as position and size, are derived from data models and used to measure data changes. For example, Samtaney et al. [114] introduced several evolutionary events and tracked 3D data according to their feature attributes. Banks and Singer [17] used a predictor-corrector method to reconstruct and track vortex tubes from turbulent time-dependent flows. Reinders et al. [109] matched several attributes of features and tracked feature paths based on the motion continuity. Verma and Pang [134] proposed comparative visualization tools for analyzing vector datasets based on streamlines. Post el al. [105] provided a good survey of flow visualization, which includes many feature-based approaches. We extend the design of feature descriptors to a set of independent local data properties. More importantly, we derive our concept of virtual words from feature descriptors as representative key local data properties.

2.3 Temporal Trend Analysis

Extracting the temporal changes from the dataset can be beneficial and help users to better understand the underlying structure. A lot of work has been done in this are. Woodring and Shen designed a global time view spreadsheet, which grouped similar activities that are clustered using wavelet transform along time [151]. Lee and Shen presented an algorithm called SUBDTW to identify trends appear and vanish in multivariate time-varying data [89]. Wang et al. presented an importance-driven method, which derived an importance curve for each data block based on the formulation of conditional entropy from information theory and clustered importance curves to visualize temporal trends [141]. Woodring and Shen also utilized temporal clustering and sequencing, creating a corresponding transfer function, to find dynamic features and describe the value range distributions over time in value space [152]. Janicke et al. described a framework based on wavelet analysis to investigate variability climate changes [76]. Turkay et al. proposed two novel and interactive visualization techniques, temporal cluster view and temporal signatures, that enabled users to explore and interpret the structural changes of temporal clusters [132]. Hao et al. introduced automated peak-preserving smoothing and prediction algorithms, enabling a reliable long term prediction for seasonal data. They also combined them with an advanced visual interface to explore seasonal patterns with new visual interaction techniques to incorporate human expert knowledge [67].

Mao proposed Lowbow (Locally weighted bag of words) algorithm for document visualization using curves [98]. Typically a document is considered as a vocabulary with a finite sequence. By treating a document as a discrete categorical time series dataset and locally averaging a word histogram at different location in the document, a local version of the global histogram can be obtained to describe local word distribution. In other words, the word histogram which is a high dimensional vector contains a probability distribution over the vocabulary at one document location. By viewing the histograms geometrically, Lowbow generates a smooth curve to summarize the progression of the semantic and statistical trends within the document.

Similarly, Assa et al. presented human motions in succinct line drawings by selecting key poses based on the analysis of a skeletal animation sequence [14]. Ward and Guo borrowed the idea in text analysis and presented an approach to time-series data visualization, creating multivariate data records out of short subsequences of the data and then using multivariate visualization methods to display and explore the data in the resulting shape space. They mapped each temporal N-gram to a glyph, and then positioning the glyphs via PCA [144]. We modify the Lowbow algorithm for time-varying data visualization to keep the advantages of Lowbow algorithms on sampling data from distribution statistics.

Lu et al. proposed an interactive storyboard approach [94] for visualizing overall data contents and relations of time-varying datasets. There are three main differences between our approach in Chapter 4 and the storyboard method [94]. First, the storyboard method compares two time steps directly based on a set of pre-selected features. Our approach compares two time steps based on the distribution of virtual words. Since the virtual words are generated with the 3D SIFT algorithm, which is invariant to feature transformation, our approach can better capture data characteristics. Second, we adopt the mechanism of Lowbow to develop a statistical time sampling method. Differently, as the storyboard approach compares volume datasets directly, all the time steps from selected time ranges are involved. Third, the storyboard approach visualizes a time-varying dataset with a specially designed storyboard, which selects representative time steps and embeds their snapshots into the time line. Our approach explores several methods to compare different time ranges or data attributes. In summary, since we represent local data properties by a set of meaningful virtual words, our approach tends to be more intuitive. Our approach also provides the flexibility for sampling time steps through a modified Lowbow algorithm, thereby it is more efficient for data with a large number of time steps.

2.4 Ensemble Visualization

Ensemble visualization has been studied extensively during recent years due to the growth of such simulation datasets. Because of the complexity of the ensemble datasets, the research area covers a large portion in both scientific and information visualization. Geo-spatial visualization, volumetric rendering, animation, multivariate, multi-dimensional data visualization, user interactivities are all included. We discussed most of those work in the previous sections.

Techniques of visualizing multi-variate and multi-dimensional datasets have been used extensively in ensemble visualization because of the similarities of the data properties. Thus, these techniques can be applied to the ensemble members directly. Due to the rich information included in the ensemble dataset, a single representation is obviously not sufficient. Interfaces of multiple, linked views of data can solve this issue. Fuchs and Hause discussed relevant research works related to the visualization of complex, multivariate data and provide a categorization of these techniques [56]. Anselin et al proposed a framework of dynamically linked windows, combining multiple representations of data on a map with traditional statistical graphics, such as histograms, box plots, and scatterplots [12]. Xie et al. proposed an approach to define quality measures for multivariate tabular data and presented several approaches to visually map quality information with information visualization techniques such as parallel coordinates and star glyphs [155].

Here we review several work of ensemble visualization that have been done. Short-range ensemble forecasting runs on a daily basis is used to predict near future weather patterns [1]. Gneiting and Raftery generated ensemble datasets which combine multiple runs of simulation using perturbed parameters and variations on initial conditions to make predictions on local weather [65]. Wilson et al. presented a general approach for gaining key scientific insight from ensemble data through a collection of overview and statistical displays [108, 107]. They proposed an interactive framework consisting of a collection of overview and statistical displays to help understanding ensembles.

They used a collection of statistical descriptors to summarize the data, and displayed these descriptors using variety of visualizations which are familiar to domain experts. Phadke et al. proposed methods for visualizing ensembles: one pairwise sequential animation technique that combines subsets of members using visibility control, and a tinting technique that presents differences between members using screen space subdivision and saturation tinting [104]. Talbot et al. proposed EnsembleMatrix, an interactive visualization system that presents a graphical view of confusion matrices to help users understand relative merits of various classifiers [127]. Sanyal et al. described a tool named Noodles which uses coordinated view of ribbon and glyph-based uncertainty visualization to visualize ensemble uncertainty [115]. Other than these method which just shows simple features in their displays, we incorporate our time line approach and generate more meaningful ensemble visualization. Also, we are capable of displaying large number of analyzed members without occlusion.

The major problem with these work is that they do not provide a method to address the porblems of clutter and occlusion, When the amount of ensemble grows, they become impractical. Our methods provide a way to reduce the visual cluttering and still be able to study the correlations between members and variables.

2.5 Animation

2.5.1 Animation in Visualization

Animation has several advantages comparing to traditional static presentations.

- Animation is able to help users to track desired features or events.
- Animation can direct user attentions to important objects.

- Animation is able to incorporate a story-telling style to describe the data property narratively.
- Animation is exciting and engaging to users.

It's been widely used in every aspect of education [125, 84], computer graphics, and visualization. Many papers have performed experiments to support the idea that animation is beneficial for both scientific and information visualization. Stasko et al. did a series on how algorithm animations could assist learning [125, 84]. Initially the result did not show significant benefits from animation. But later animation was found useful when they changed the environment and materials and added interactions with the animations.

Many visualization systems provide a certain degree of animation support, ranging from recording keyframes to producing animations automatically. For example, Akiba et al. [11] developed a template-based animation tool for volume visualization. Gershon [61] presented methods for visualizing fuzzy data, including displaying a series of blurred images in an animation loop. Viola et al. [135] presented a method to focus viewpoints automatically on features of a volumetric dataset. Other types of animation include animation of 2D steady vector fields [90], animation of orthogonal texture patterns for vector field visualization [15], and 3D interactive animation in information visualization [112].

Animation has also been applied to enrich static visualization, providing animation support to assist user change perspectives, highlight and track objects [95, 150, 37]. For example, Lum et al.[95] presented kinetic visualization, which visualized animated particles over an object surface to enhance the visual perception. Correa and Silver [37] produced animations that highlighted data features by traversing the volume along a path specified via transfer function. Woodring and Shen [150] highlighted objects in a volume dataset by applying animations with positional motions and opacity variations. Also, Blumenkrants et al. [22] created narrative algorithm visualization with the algorithm graph based on information like a central plot or a story. Results of their study indicated that the narrative visualization appeared effective.

Different from previous approaches of animations, our approach in Chapter 6 generates animations automatically by simulating storytelling techniques and integrates automatic composition algorithms to produce smoothly transitioned animations. Similar to the multiresolution video method [53], our approach allows animation generation with different levels-of-detail. Our approach is also relevant to the algorithm visualization method [22] on building narrative structures through tree search strategies.

2.5.2 Storytelling in Visualization

Storytelling technique in visualization has been seldom studied. Gershon and Page [60] discussed the usage of stories in information visualization, especially for cases when data characteristics were abstract and could not be visualized in the form of a picture in a satisfying way. Wohlfart and Hauser [148] presented a method to use storytelling to represent a volume dataset through the processes of story authoring and story telling. They demonstrated the potential of their approach with medical visualization examples. Similarly, our approach in Chapter 6 can also use animations to represent data information. Different from previous approaches, our approach exploits the automatic generation of animations for time-varying datasets through the detection of event features and construction of narrative structures. Our approach also allows semantic visualizations, which are more flexible and meaningful for domain users.

2.5.3 Evaluation of Animation

Animations are used more often in information visualization area for visualizing transitions between views and datasets, presenting overall data trend, and illustrating processes along the time, for example dynamic graphs and networks [52]. The effectiveness of animation is closely related to the data type. Tversky and Morrison [133] found out that animation may be ineffective when displaying events and were often too complex and too fast to be accurately perceived. However, they still acknowledged that animation allowed interactions such as close-ups, zooming, alternative perspectives and control of speed, which were likely to facilitate perception and comprehension.

There has been a lot of work evaluating the effectiveness of animation for different purposes in the information visualization area. Heer and Robertson proposed design principles for creating animated transitions for data graphs and performed user studies finding that animated transitions significantly improved graphical perception [70]. Robertson et al. [111] evaluated the effectiveness of animation in trend visualization and draw the conclusion that small comparable visualization was the most effective approach. They discovered that animation worked well in presentation tasks but not as good as other techniques for analysis purposes. All the techniques have their own advantages for particular situations. However, Lundström et al [96] presented an animation method to convey uncertainty in medical visualization and evaluated their method by employing radiologists in a study simulating the clinical task of stenosis assessment, in which the animation technique was shown to outperform traditional rendering in terms of assessment accuracy. Boyandin et al. presented a qualitative user study analyzing findings made while exploring changes over time in spatial interaction with flow maps using animation and small-multiples as two alternative ways of representing temporal changes. They concluded participants with animation tended to make more findings concerning geographically local events and changes between subsequent years; while with small-multiples there are more findings concerning longer time periods were made [25]. We believe that for data with meaningful 3D structures, like our time-varying datasets, animation can visualize temporal events effectively.

We evaluates the effectiveness of animation in 3d data visualization serving different purposes in Chapter 7. We examine if animation is beneficial under different visualization tasks. We also performed a formal user study comparing our semantic animation in Chapter 6 to interactive visualization with exploration and analysis tasks.

CHAPTER 3: APPLICATIONS

In this chapter, we introduce the simulation large scale datasets we used in this dissertation. The datasets include volumetric data and 2D simulation data on grids. Some of the data are results of forecasts from complex simulation models.

3.1 Time-varying Volumetric Data

Volumetric data has found extensive use in geological and medical applications such as 3D CAT scans and MRI's. We mainly uses three time-varying volume dataset in this dissertation. Each of these datasets is in 3D with multi-variate along the time. We mainly use three volume datasets as below. Figure 1 shows example renderings of each dataset.

1. Air Quality Air quality data are generated from the community multi-scale air quality (CMAQ) [2] and the sparse matrix operator kernel emissions (SMOKE) [3] modeling systems. SMOKE is also one input source of the CMAQ system, which simulates various chemical and physical processes that are important for understanding atmospheric transformations and distributions. The chemical pollution are simulated across North America including chemicals such as NH3, SO2, NO2 and NO. There are 25 timestep each day and the dataset consists of 365 days which is a total of 9125 volumes. The size of each volume is 112x148x19.

2. Jet Engine Jet engine dataset simulates the process of engine changes of a jet. It has two simulation variables: density and energy. Each variable has 200 timesteps and each volume data is 128x128x128.
3. Turbulence The turbulence data is mathematica turbulence simulation which consists of 4 different conditions and 249 volumes under each condition. The size of each volume is 100x100x100.

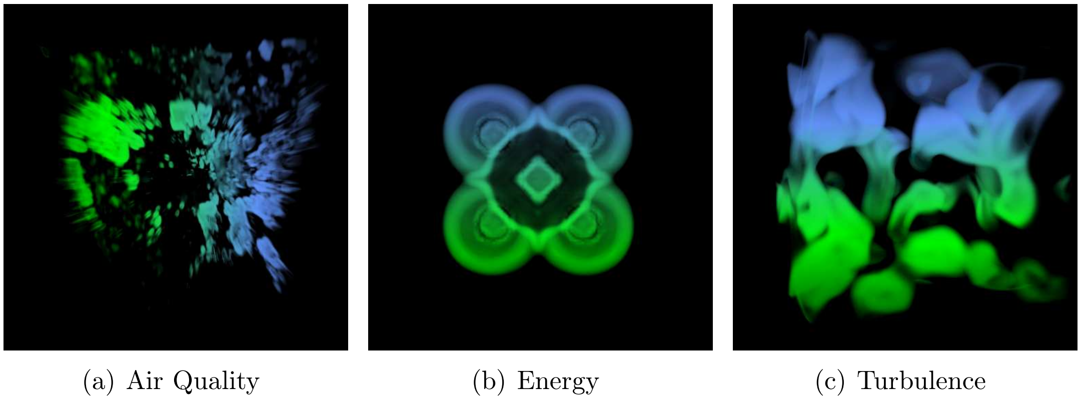

(a) Air Quality (b) Energy (c) Turbulence

Figure 1: Example volume renderings of different datasets.

3.2 Coastal Circulation and Storm Surge Model

Besides volumetric datasets, we also use 2D or 3D simulated storm surge or oil particle datasets. These dataset are simulated by a coastal circulation and storm surge Model named ADvanced CIRCulation (ADCIRC) model [4]. ADCIRC is a system of computer programs for solving time dependent, free surface circulation and transport problems in two and three dimensions. These programs utilize the finite element method in space allowing the use of highly flexible, unstructured grids [41, 44]. Typical ADCIRC applications have included: (i) modeling tides and wind driven circulation, (ii) analysis of hurricane storm surge and flooding, (iii) dredging feasibility and material disposal studies, (iv) larval transport studies, (v) near shore marine operations. It has been employed successfully to coupled wind, wind-wave, tide and riverine ow simulations on unstructured meshes in many geographical regions including the Gulf of Mexico. ADCIRC has been coupled recently to the Simulating WAves Nearshore (SWAN) model, so that both models run on the same unstructured meshes and on the same computational cores [162, 44]. The resulting SWAN+ADCIRC model is well-positioned to simulate accurately and efficiently the propagation of wind-waves, tides and storm surge from deep water onto the continental shelf and into the nearshore [42, 35].

Simulations are usually in a different format. The simulation result files from ADCIRC are in NetCDF format which is a set of software libraries and self-describing, machine-independent data formats and array-oriented scientific data format [5]. We download all the daily simulations and use NetCDF libraries to interpret the ensemble dataset. Each ensemble member contains more than 10 variables including 1D scalar values or 2D vectors such as elevation values and wind vectors. We select simply features directly from these as well as calculating inundation area, wind speeds and other surge related features. The oil spill dataset only contains the geo-location information of each particle over the time.

3.2.1 Surge Simulation

The surge simulation has two categories, one is long time duration containing one major storm, the other is ensemble surge simulation with short time duration but numerous of runs. We introduce more details of storm simulations with their own features in Section 7.2.2.1. Both of the ensemble and storm simulation share the same format.Different from the storm surge dataset, ensemble simulations are relatively short in time and most of them do not include tropical cyclone. The surge simulation are produced several time a day and in great number. These daily runs are used for many different things, and different clients will be looking for different things. Some may be interested in the tides, others will be interested in the wave forecast, etc. Marine scientists are producing a wide variety of data that will be used differently by different audiences.

The simulation lies on a grid which consists of 295,328 vertices and 520,114 triangles. Figure 2 (left)shows the mesh grid of the surge simulation. Each simulation contains 84 timesteps, which means 84 hours, and more than 20 variable outputs are produces such as sea surface elevation, atmospheric pressure, sea water velocity, wind velocity, peak wave period, mean absolute wave period, significant wave height, mean wave direction and so on. Among these variables, sea surface elevation and wind velocity are the most important and interesting ones. Figure 2 (right) shows rendering examples of these parameters.

3.2.2 Oil Spill Simulation

Beyond natural disasters, accidents can create serious problems too. The destruction of the Deepwater Horizon drilling platform during the spring of 2010, which put the northern Gulf of Mexico in threaten by an oil spill, also raised concerns of emergency responders. The oil spill posed a serious environmental threat to northern Gulf coastline from Florida to Texas. There was also concern that oil would be carried by

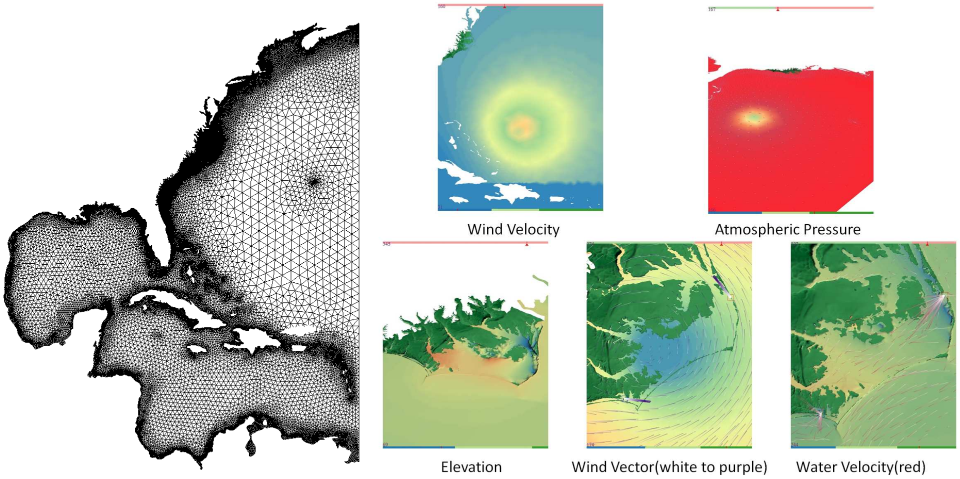


Figure 2: Left: Mesh of the surge dataset. Right: Sample renderings of different variables in the simulation.

the currents to more distant regions. Early predictions showed oil could make its way to the Atlantic Ocean and the eastern seaboard of the United States. Other concerns such as the potential of a major storm could impact the northern Gulf during the spill.

Due to these facts, scientists generate simulations to study how oil behave under different circumstances and parameters. The oil spill simulation consists of longitudes and latitudes information of over 10 million particles. The major parameters changed between different simulations are winds and currents [43].

As shown in Figure 3, three simulations are produced with the same time duration but under different parameters. The particle locations are sampled and connected across different time steps to show the trajectories moving from orange to red. The trajectories show a general movement of the oil spill to the north and east. In Figure 3 (a), the particles are pushed only with currents while the other two added different wind forcings. When only the currents are applied as forcing, the particle motion

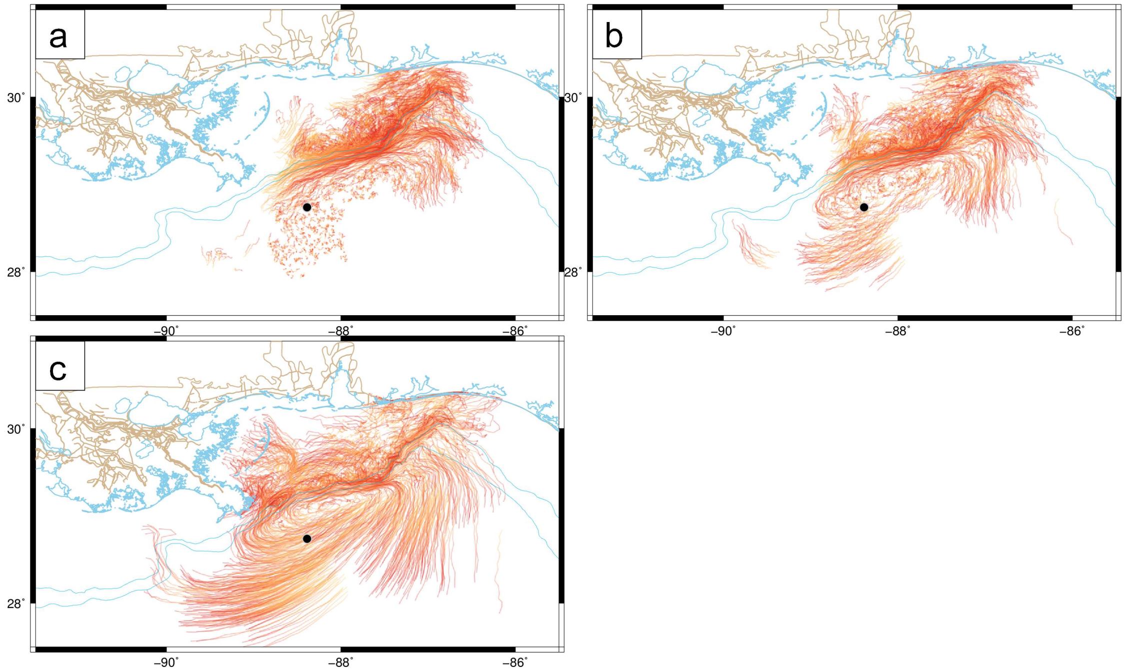


Figure 3: Sampled predicted particle locations in time during the mid-June period, with forcings from (a) currents only, (b) currents and 1% winds, and (c) currents and 3% winds. Particle movement is shown with the trajectories moving from orange to red. The platform location is indicated.

is limited mostly to the continental shelf, with the particles nearly stationary in deepwater. This behavior is supported generally by the observations. Figure 3 (b) and (c) show when winds are included as forcing, the oil particles have a very different trajectories, moving too much in deepwater, especially to the south and west. Figure 4 shows a different simulation sets under the condition of simulated storm effects. As we can see, scientists produce tons of simulations to observe the different behaviors. The results can tell them what the appropriate parameters are and make predictions when real life scenarios happen.

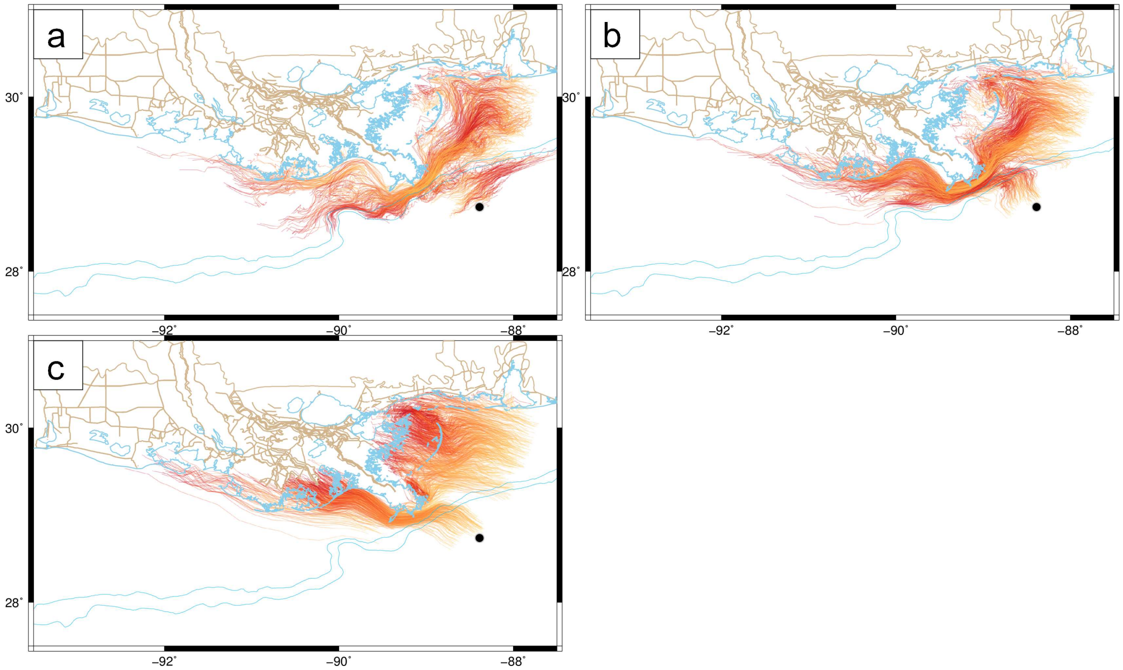


Figure 4: Sampled predicted particle locations in time during the Alex, with forcings from (a) currents only, (b) currents and 1% winds, and (c) currents and 3% winds. Particle movement is shown with the trajectories moving from orange to red. The platform location is indicated.

CHAPTER 4: DESIGN AND GENERATION OF TIME LINE

4.1 Introduction

In this chapter, we present an approach to construct a space of various data properties and extract corresponding virtual words. We define “virtual words” as data properties, in the form of high-dimensional vectors, at feature locations in the 3D space. The virtual words are further used to generate succinct time lines for analyzing and summarizing time-varying datasets. We concentrate on visualizing scalar, multi-field, and time-varying datasets, although the concept of virtual words can be used to study general temporal events.

Our approach is derived from the fact that human languages can effectively describe all kinds of objects and events in the world with a limited set of words. For example, linguistics specialists Berlin and Kay [19] have identified eleven possible “basic color" categories, such as red, green, and yellow, from a study involving twenty different languages. As shown in Figure 5, the rest color in the continuous 3D color space is often described according to their similarities to the basic color, e.g., we can say a color is somewhat blue. Similarly, for time-varying datasets, we collect data values from various feature aspects to build a data property space, corresponding to the continuous color space. Representative locations from this space are selected, corresponding to the basic colors. We define the representative locations in the data property space as “virtual words", which are actually high-dimensional vectors encoding different data features. With the collection of virtual words, arbitrary data properties can be described according to their locations in the data property space. In addition, the transitions of locations in the data property space can be used to represent data changes across time. When data properties from different time steps vary, we can measure their relationships through the distribution similarities of corresponding virtual words.

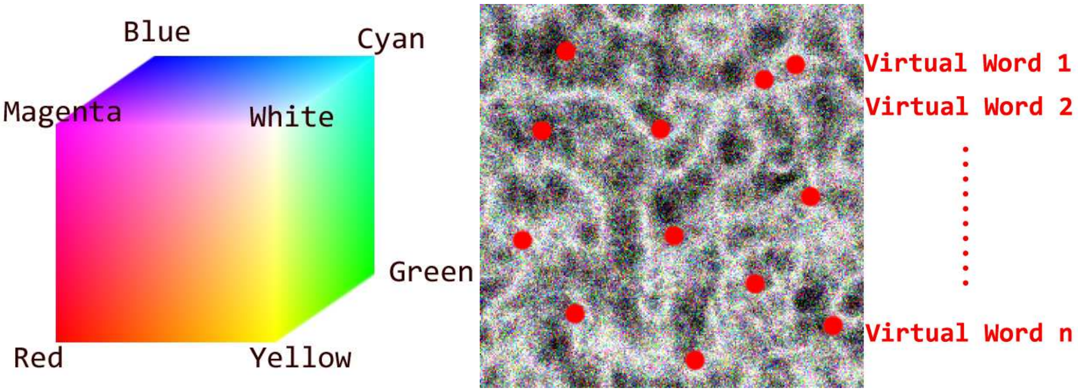


Figure 5: (Left) Only several words exist to describe the entire 3D color space. (Right) Similarly, we can build a space of data properties and extract virtual words to describe data and data relationships.

As shown in Figure 6, our approach consists of several stages. First, we select feature points by means of detecting extrema locations in both the datasets and object boundaries for individual time steps of a time-varying dataset. Around these feature points, we additionally generate feature descriptors by collecting various local data properties. Second, we extract a set of virtual words as high dimensional vectors through clustering in the space of data properties. We further categorize all the feature points according to their distances to the virtual words. Third, we use the distribution statistics of virtual words from different time steps to produce meaningful time lines with a modified method of locally weighted bag of words (Lowbow) [98]. Fourth, we present several time line visualization methods and demonstrate with examples that our approach can be used flexibly to explore and compare time-varying datasets.

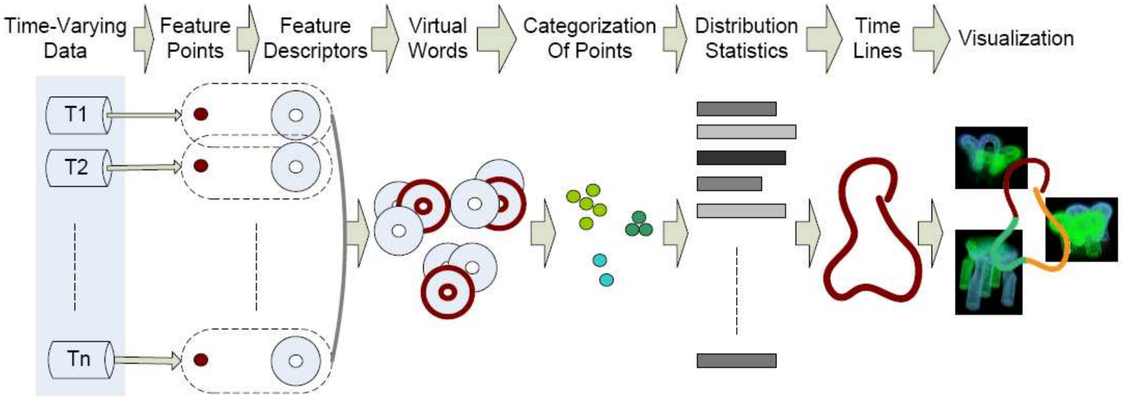


Figure 6: Our approach consists of stages to select feature points and feature descriptors, extract virtual words, and sample their distribution statistics across time to produce meaningful time lines for studying temporal relationships.

The main contribution of the time line approach is the novel means of generating time lines based on extracted virtual words for characterizing features of data properties. Our approach to constructing virtual words provides a flexible framework for users to choose their desired data properties by combining a set of independent feature descriptors. We modify the Lowbow locally weighted bag of words (Lowbow) [98] algorithm to suit the needs of time-varying data visualization and extend it to incorporate existing knowledge or hypothesis into the process of time line generation. Our approach ensures that the combination of time lines and feature descriptors provides a succinct visualization tool for data exploration, comparison, and summary when studying temporal relationships.

Partial of the work in this chapter is published in [158].

4.2 Generation of Virtual Words

This section presents our approach to extracting virtual words and generating time lines. A time line visualizes time steps as points and adjacency relationships of time steps as lines. For efficient data analysis, we adopt the mechanism from Lowbow algorithm to sample time steps through collecting statistical feature distributions. Since the locations of sample time steps are calculated carefully based on data properties selected by users, time lines can be used to analyze temporal relationships.

As shown in Figure 6, our algorithm contains three stages. First, for each time step, we select feature points and collect feature descriptors around them. Second, we cluster feature descriptors as high dimensional vectors to produce a set of virtual words that can represent local data properties. We further categorize all the feature points according to virtual words. Third, we present a modified Lowbow algorithm to use the distribution statistics of virtual words to generate time lines. The following describes the details of these three stages respectively.

4.2.1 SIFT Feature Descriptor

The SIFT algorithm is widely used in computer vision to detect and describe local features in images.

The algorithm includes four steps:

1. Scale-space extrema detection: The first stage of computation searches over all scales and image locations. It is implemented efficiently by using a difference-of-Gaussian function to identify potential interest points that are invariant to scale and orientation.

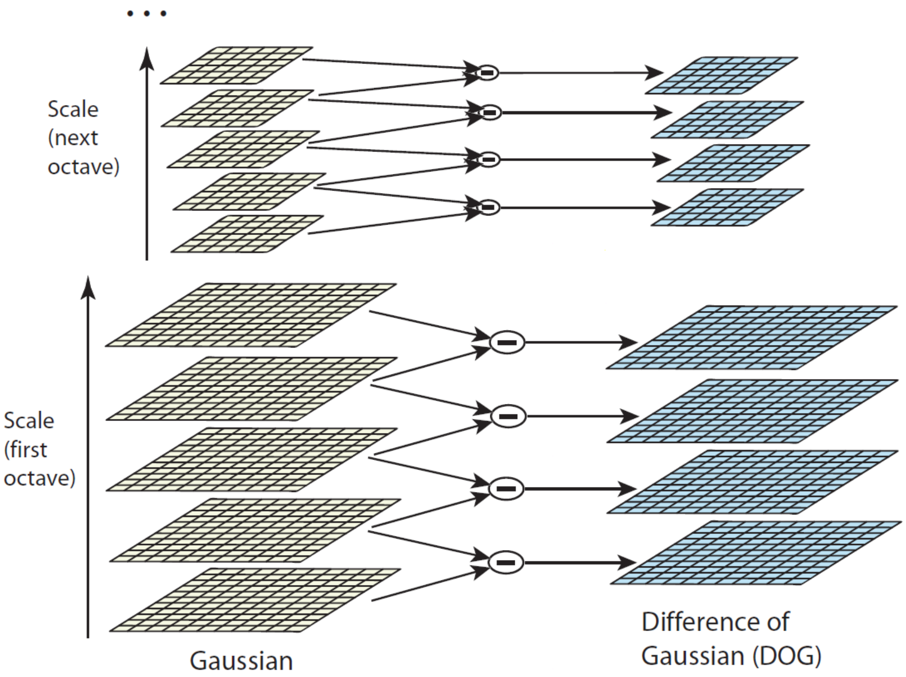


Figure 7: This figure show the image pyramid in scale space. After the DOG images are calculated, we can find the local extrma by comparing its 26 neighbors. In 3D SIFT, we need to compare its 80 neighbors.

2. Keypoint localization: At each candidate location, a detailed model is fit to determine location and scale. Keypoints are selected based on measures of their stability.
3. Orientation assignment: One or more orientations are assigned to each keypoint location based on local image gradient directions. All future operations are performed on image data that has been transformed relative to the assigned orientation, scale, and location for each feature, thereby providing invariance to these transformations.
4. Keypoint descriptor: The local image gradients are measured at the selected scale in the region around each keypoint. These are transformed into a repre-

sentation that allows for significant levels of local shape distortion and change in illumination.

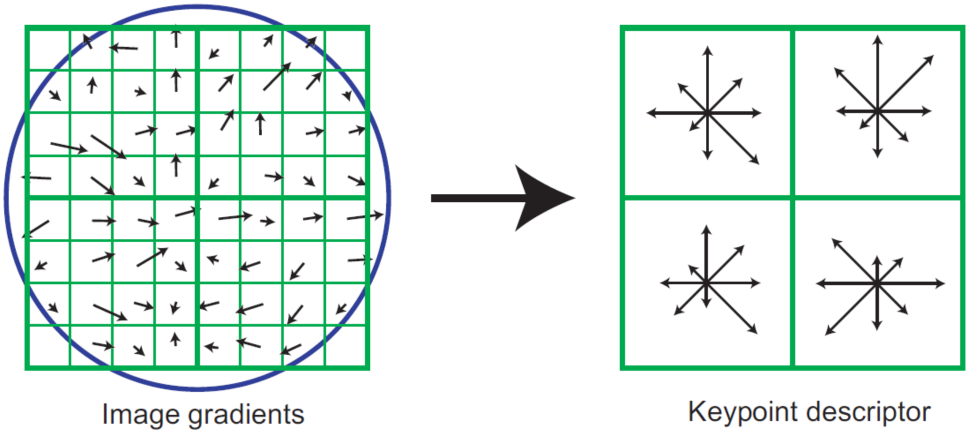


Figure 8: A keypoint descriptor is calculated by computing the gradient magnitude and orientation at each image sample point in a region around its location, as shown on the left. These are weighted by a Gaussian window, indicated by the overlaid circle. These sampled gradient magnitude and orientation are then accumulated into orientation histograms summarizing the contents over 4x4 subregions, as shown on the right side, with the length of each arrow corresponding to the sum of the gradient magnitudes near that direction within the region. This figure shows a 2x2 descriptor array computed from an 8x8 set of samples.

4.2.2 Selection of Feature Points

While the definition of feature points varies, we combine two criteria: one is the local extreme locations from volumetric SIFT algorithm [118], and the other is boundary points. Since feature points are selected independently for each time step, this approach potentially allows different selection criteria or feature extraction methods to be applied for different time steps.

We define "virtual words" as data properties, in the form of high-dimensional vectors, at feature locations in the 3D space. The first two steps of our approach are to select feature locations and to collect feature descriptors. We build our approach based on the SIFT algorithm, as it provides robust feature tracking which is invariant to rotation, scaling, noise or changes in illumination. These characteristics make it appropriate for extracting features from volume datasets. Later, we describe our method to generate a set of virtual words from the collected information. The following describes the details of these three stages respectively.

For 3D scalar datasets, we use volumetric SIFT algorithm to select feature points with two main steps: scale-space extrema detection and point filtering. Simply speaking, the first step selects a set of candidate points and the second step removes some insignificant candidate points.

In the first step, a scale space of difference-of-Gaussian (DOG) is built as follows. We first scale a volume to several different levels: ×2, original size, /2, /4, etc. Similar to the method introduced in [102], these volumes are convolved with a 3D Gaussian filter $G(x, y, z, k\alpha)$ of n different scales α, $k\alpha$, $\cdots$, $k^n\alpha$ to generate a scale space. A volume pyramid is built by subtracting the differences between adjacent scale spaces. The choice of k is relevant to the scale of data features, as larger k^n results in smoother effects. For all the results here, we use 4 volume levels and 12 DOG scales ($n = 12$), and $2^{1/12}$ for k. From this scale space, we automatically select a set of candidate points by detecting local extrema locations. Generally the DOG value of each candidate point is compared to its 80 neighbors, of which 26 in the current volume, and 27 in each of its two adjacent volumes on the same level. A location is selected as a candidate only if it is larger than all of its neighbors or smaller than all the neighbors.

The second step is to filter out inappropriate candidates that are poorly localized along an edge or with low contrast. As described in [102], a poor candidate feature point has a high principal value along the edge direction but a small value in the perpendicular direction. We calculate the curvature values $\lambda_1, \lambda_2, \lambda_3$ using the method from [66], where $\lambda_1 < \lambda_2 < \lambda_3$. A candidate point is valid if λ_3/λ_1 is smaller than a user specified threshold. Larger thresholds can produce more feature points on the edges. The edge response in 3d space can be measured by a 3 x 3 Hessian matrix,

$$H = \begin{bmatrix} D_{xx} & D_{yy} & D_{zx} \\ D_{xy} & D_{yy} & D_{zy} \\ D_{xz} & D_{yz} & D_{zz} \end{bmatrix} \tag{1}$$

where D_{ij} is the second derivatives computed at the location and scale of the feature points. The eigenvectors define an orthogonal coordinate system aligned with the direction of minimal (e1) and maximal (e3) curvature. Let γ be ratio between the largest eigenvalue and the smallest one, so that $\gamma = \lambda_3/\lambda_1$. The trace and determinant of the matrix H can be calculated by the following equation:

$$Tr(H) = \lambda_1 + \lambda_2 + \lambda_3 = D_{xx} + D_{yy} + D_{zz} \tag{2}$$

$$\begin{aligned} Det(H) &= \lambda_1\lambda_2\lambda_3 \\ &= D_{xx}D_{yy}D_{zz} + 2D_{xy}D_{yz}D_{xz} - D_{xx}D_{yz}^2 - D_{yy}D_{xz}^2 - D_{zz}D_{xy}^2 \end{aligned} \tag{3}$$

If we define $\alpha = \lambda_2/\lambda_1$, then

$$\frac{Tr(H)^3}{Det(H)} = \frac{(\gamma + \alpha + 1)^3}{\gamma\alpha} \tag{4}$$

The quantity $\frac{(\gamma+\alpha+1)^3}{\gamma\alpha}$ is at a minimum when the three eigenvalues are equal and in increases with γ and α. Therefore, the feature points is accepted when $\frac{Tr(H)^3}{Det(H)} <$

$\frac{(\gamma+\alpha+1)^3}{\gamma\alpha} < \frac{(\gamma+\gamma+1)^3}{\gamma^2} < \frac{(2\gamma_{max}+1)^3}{\gamma_{max}^2}$. We use 15 as the threshold of γ_{max} for all our results based on observation.

We then sort all the rest of candidate points by their values in the scale space and allow users to choose the number of points. In our experiment, we provide users a tool to visualize feature points and 3D data directly, so that users can adjust this parameter easily. As shown in Figure 10, both points and volumes are colored according to their locations on one axis (Y axis is used here) to improve the understanding of their spatial relationships.

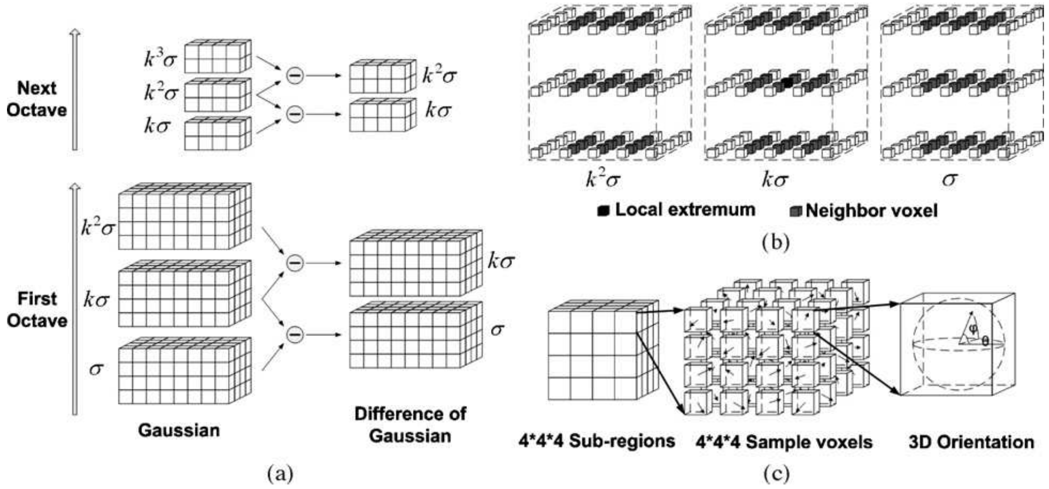


Figure 9: Volume Pyramid and 3D SIFT Feature Descriptor. A candidate point is found by comparing to its 80 neighbors in DOG space, 26 in the same octave and 27 each in the 2 neighbor volumes in the same octave. A 3D SIFT feature descriptor is created by calculating the 4x4x4 sub-regions surrounding each feature point sampling in the rotated neighborhood. Each sub-region, the magnitude of the gradient, weighted by a Gaussian window centered at the feature point, is added to the corresponding bin for the gradient orientation, where 8 bins are used for θ and 4 bins are used for ϕ. [102]

For 3D datasets, users often choose objects-of-interest, e.g., through adjusting transfer function. Thus the above selection process may miss the important information of object boundaries. We accordingly compliment the volumetric SIFT algorithm by considering the influence of boundary locations. Specifically, when users adjust transfer functions, we modify the volume pyramid by removing all regions that are out of objects-of-interest. These regions are not considered during the process of scale-space extrema detection. This simple modification can ensure that all the feature points inside objects-of-interest are preserved and extra points at the object boundaries will occur. We accelerate the process of feature point selection by precomputing all feature points and building the volume pyramid. During interaction, we modify the volume pyramid according to newly selected feature points around the boundaries of objects-of-interest.

Figure 10 shows the results of interactive selection of feature points. The 2D transfer functions of voxel data values and gradient magnitudes [86] are used. The top left image shows all the feature points when the entire histogram is selected. The other three images demonstrate the results of point selections when different transfer functions are determined.

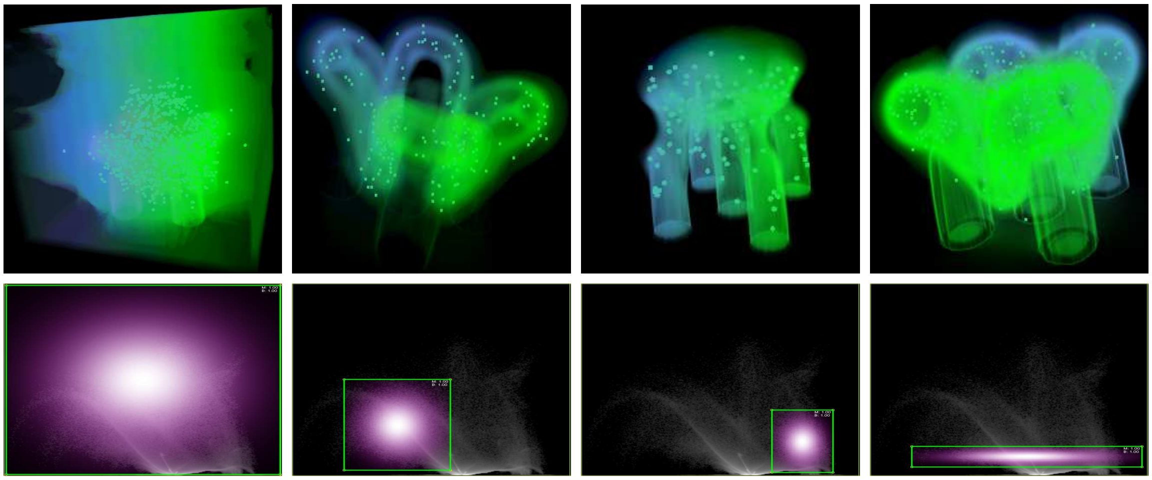


Figure 10: Selection of feature points (shown on the first row) according to the choices of transfer functions (shown on the second row).

4.2.3 Collection of Feature Descriptors

We extend the feature descriptors presented in the original SIFT algorithm [92] to describe various local data properties. Different from the original SIFT algorithm, the interests of data exploration include the rotations and movements of objects. Therefore, we add the components of gradient orientation for rotations and the object location for movements in the feature descriptor. Second, we add components from texture analysis measurements for describing local data properties. Since each component describes a different data property, our construction method ensures the independency among all the components of feature descriptors. This allows users to freely combine these components to describe the data properties under exploration.

The advantage of the feature descriptor derived from the standard SIFT algorithm is that it is invariant to orientation, scale, and location of feature points. Simply speaking, it assigns one orientation to a feature point based on local volume gradient directions and performs the rest procedure on transformed data. We utilize this advantage of SIFT algorithm to generate multiple independent components of feature descriptors.

Specifically, the first component, gradient orientation, is built as follows. Assume we represent the gradient orientation in 3D by two angles, θ and ϕ. For each feature point, we rotate its neighborhood region to the direction of $\theta = \phi = 0$ when computing its feature descriptor. The rotation matrix is defined as

$$R = \begin{bmatrix} \cos\theta\cos\phi & \sin\theta\cos\phi & \sin\phi \\ -\sin\theta & \cos\theta & 0 \\ -\cos\theta\sin\phi & -\sin\theta\sin\phi & \cos\phi \end{bmatrix} \tag{5}$$

With this matrix, we sample a $4\times4\times4$ sub-region surrounding the feature point in the rotated neighborhood. At each voxel of this sub-region, we generate a feature vector according to θ and ϕ. To quantify the value space, we use 8 bins for θ and 4 bins for ϕ. The gradient magnitude, weighted by a Gaussian window function centered at the feature point location, is added to the corresponding bin for the gradient orientation. In this way, the component of gradient direction has $4\times4\times4\times8\times4 = 2048$ dimensions. Our approach allows users to select larger values, which can produce more accurate results but take longer computation time. We choose the described parameters by balancing these two factors.

We use the vector (θ, ϕ) as the second descriptor component and the location vector (x, y, z) as the third descriptor component. This is necessary for time-varying data visualization, otherwise when an object moves or rotates across time, the time line will be just a point. We also add the time step when a feature point occurs as the fourth descriptor component, which will be used in the generation process of time lines in Section 3.4.

We have also explored other factors that can benefit time-varying data visualization. We mainly apply the first order of statistics for texture analysis to collect information of local data properties, such as variance and histogram [29]. Note that since this information is collected on the data after the rotation, it is independent to the gradient orientation. Obviously, it is also independent to location and time steps. Therefore, we can arbitrarily choose a combination of these descriptor components. The generation of time lines described below in Section 3.4 also allows users to select different weights for each descriptor component.

Figure 11 shows the time line results from different feature descriptors for a time-varying energy dataset. Overall, the time lines suggest that the data changes gradually during the initial period (rendered in blue to cyan) and data properties are similar during the middle and end durations (rendered in yellow to red). We can see that the resulting time lines can be very different for these descriptor components representing different data properties. Therefore, it is important to provide users both the results of time lines and the choices of feature descriptors, so that they are aware of the meanings of time lines.

4.2.4 Extraction of Virtual Words

From the collection of feature descriptors, we extract virtual words as follows. Here we use the “applicable components" of feature descriptors to refer to all the components except the location. Our first step is to cluster each applicable component of feature descriptors independently. This clustering step can be viewed as selecting important virtual words from the space of feature descriptors. This is from the fact that the number of different feature descriptors is often limited. One reason is that it is bounded by the dimensions of feature descriptors. The other is that the type of local data properties from a time-varying dataset is also limited. Specifically, we

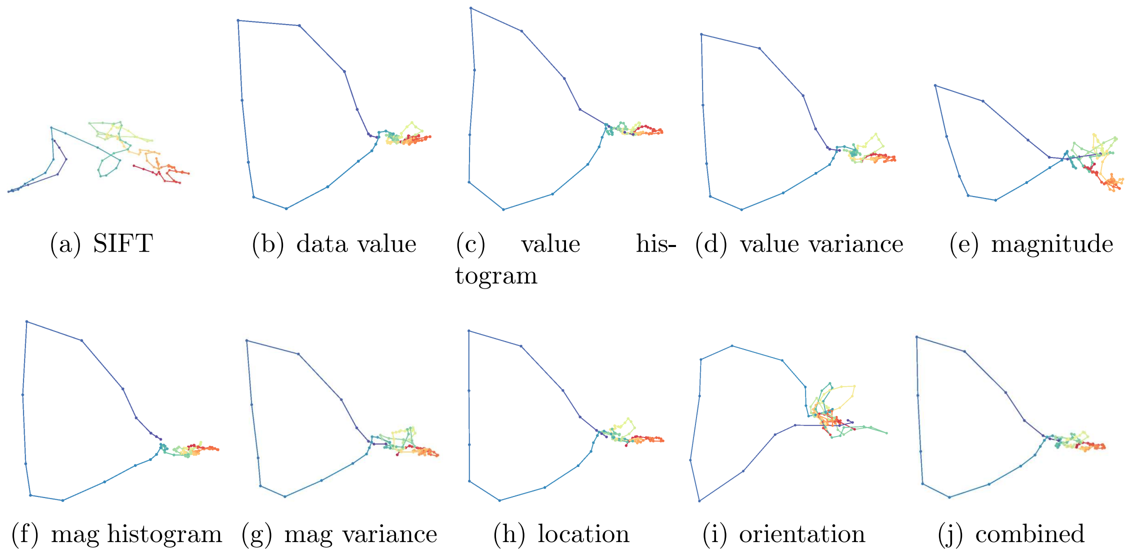
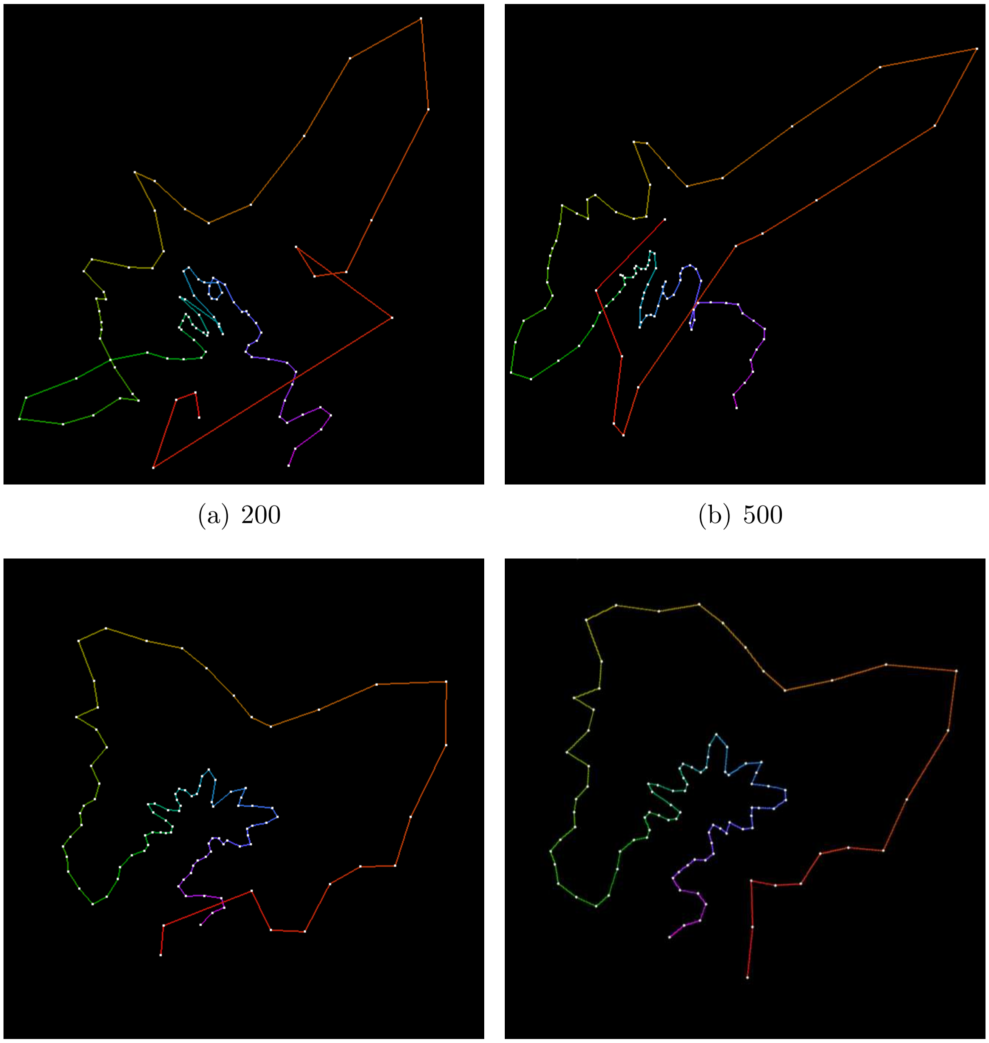


Figure 11: Time lines generated from different feature descriptors for a time-varying energy simulation dataset. Different feature descriptions yield different shapes of time lines. The first row shows the results of the sift descriptor, voxel value, histogram of value, magnitude of value, and gradient magnitude. The second row shows the results of histogram of gradient magnitude, variance of gradient magnitude, location, gradient direction, and the average combination of these nine descriptors. Overall, the time lines suggest two different data portions: the initial period (rendered in blue to cyan) and the end period (rendered in yellow to red).

use K-means [34, 59] as the clustering method since it is flexible to operate on any dimension. The choices of the clustering numbers should be different for various cases. From our experiments on both synthetical and real time-varying datasets, we find that the results are very similar as long as the cluster number is large enough. Also, the statistical prospective of Lowbow algorithm used in Section 3.4 values large sampling size. Therefore, we use 15 times the average number of feature points per time step as our clustering number. As shown in Figure 12, the results of 1000 and 1500 clusters are similar and close to the real scenario. In this way, a set of virtual words is selected for each applicable component of feature descriptors.

Figure 13 provide examples of automatically extracted virtual words. Since virtual



(a) 200

(b) 500

(c) 1000

(d) 1500

Figure 12: Results of different clustering numbers show that the time lines are similar for 1000 and 1500 clusters.

words are vectors in the high dimensional space of data properties, they are hard to visualize directly. We illustrate them in Figure 13 by marking their locations in the volume. The counterparts of virtual words are the feature descriptors at these locations. The colors represent different virtual words. We can see that virtual words often appear at representative locations with large gradients or on the object boundaries. In Figure 14, the automatically selected feature points are the eight corners of the cube. They are also the virtual words that describe several different types of temporal events. This result is consistent with our understanding of the cube dataset.

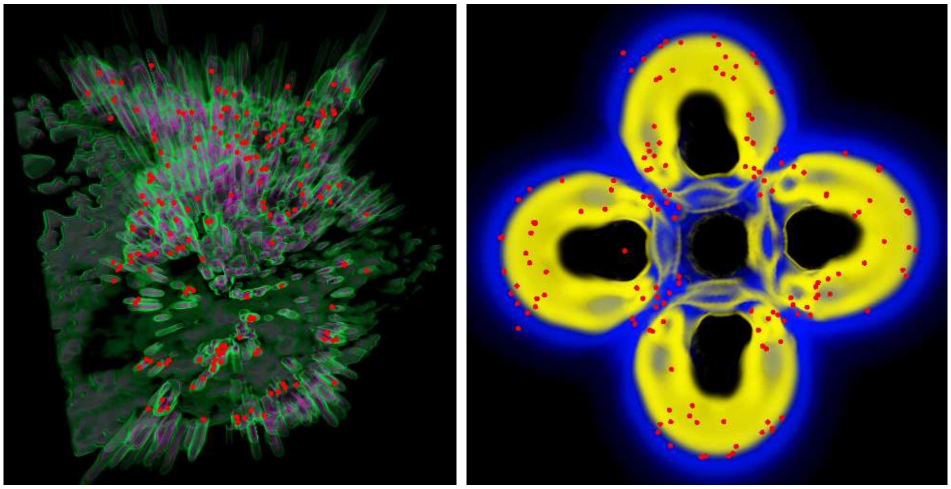


Figure 13: Example locations of virtual words, shown as the red points.

All the feature points are then categorized according to virtual words. For each applicable component of feature descriptors, we use the closest virtual word (determined by the distance in the space of feature descriptor) to represent each feature point. This is actually achieved from the clustering process simultaneously.

4.3 Creation of Time Lines

Since the virtual words cannot be visualized directly, we transform the relationships among virtual words to time lines, which visualize time steps as points and adjacency relationships of time steps as lines.

This transformation provides a visual analytics tool to study temporal evolutions. It is also a mechanism allowing flexible sampling for different requirements of performance and accuracy. For this purpose, we adopt the mechanism from Lowbow algorithm to sample time windows.

We achieve this goal with the following three steps. First, histograms are generated to record the distributions of virtual words for each time step. Second, we use these histograms to calculate dissimilarity matrices of different time steps. We also use time windows to group several timesteps into one histogram for sampling purposes. Third, time lines are produced by feeding the resulting dissimilarity matrices to MDS method [130, 24]. We project the Lowbow representation to the 2D plane, since 2D lines are more intuitive to understand.

Figure 14 shows the time line results of a synthetical time-varying data of a cube with different movement patterns and shape changes. We often use colors to indicate the temporal sequence. Most of our results use a blue to red colormap corresponding to the start to end of a time duration. As shown in Figure 14, the automatically selected feature points are the eight corners of the cube. They are also the virtual words that describe several different types of temporal events. The time lines visualize a variety of temporal events, including object movement, faster movement, object scaling, shape deformation, and effects under noise. The dissimilarities between each time steps are changing linearly in both scaling and deformation, therefore the changes of these two movements are similar. By observing the distance of points in the two time lines of Figure 14(a) and 14(b), user can see the changes of these two movements are different.

One advantage of Lowbow is the sampling feature. It allows us to sample time steps arbitrarily and emulate their histograms using different Gaussian windows. Each window covers several time steps. To ensure the smoothness of a time line, we can overlap

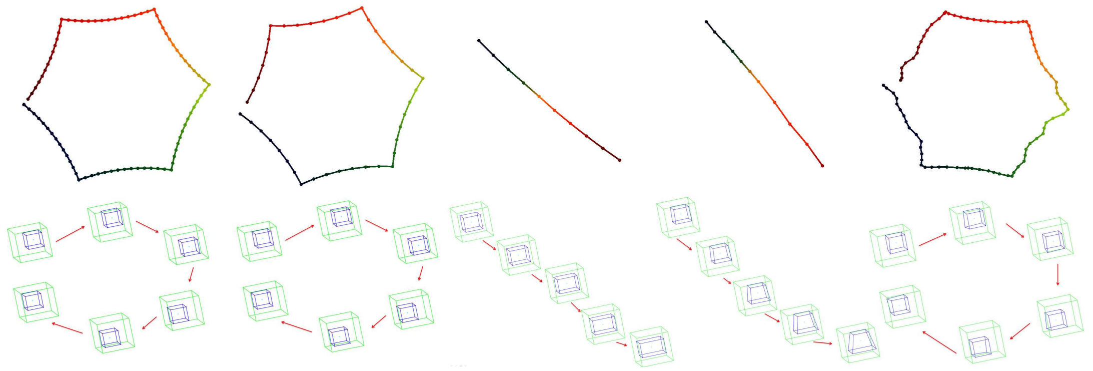

(a) Moving (b) Moving faster (c) Scaling (d) Deformation (e) Noise

Figure 14: Time lines can describe various types of temporal events. The illustrations on the second row contain several snapshots from key time steps and the time lines are shown on the first row. Different Cube synthetical Datasets are used from (a) to (e): (a) and (b) cube moving around in the 3D space along the same path at different speed, (a) cube moving at 1 voxel per timestep, (b) cube moving 2 voxels per timestep. The densities of points in the time line reflect the speed of the movement. (c) scaling of the cube, (d) deformation of the cube, and (e) cube moving around under noises. The moving paths of (a), (b), and (e) are the same, which can be detected with time lines easily.

adjacent time windows. For each applicable component i of feature descriptors, Lowbow is used to generate a dissimilarity matrix DM_i. We then combine all these matrices to a final dissimilarity matrix DM_{final} as the MDS input. Assume ST_1 and ST_2 are sample time windows. Here we allow users to assign weights $w(i)$ to the component i of feature descriptors, so that the effects of different data properties can be adjusted. When we assign larger weights for some descriptors, the combined time line is more affected by these factors.

$$DM_{final}(ST_1, ST_2) = \sum_i w(i) \times DM_i(ST_1, ST_2) \tag{6}$$

Specifically, for each applicable component of feature descriptors, one histogram is generated for each sample time step by collecting the distribution of all the virtual words in the sampling window. The values of each feature descriptor component are calculated independently for individual time steps. They are further collected for each sample time step. Some feature component only contains one value, such as the data value component (the average data values of virtual words are calculated for each time step); while the other feature components are high dimensional vectors, such as the components of SIFT and histogram of data values. Let us define $\overrightarrow{f_i(t_p, v)}$ as the value of component i of virtual word v at time t_p. Then, we can generate a dissimilarity matrix for the sample time steps by calculating the Euclidean distances of their values $\overrightarrow{f_i(t_p, v)}$. The difference value $DM_i(ST_1, ST_2)$ of two sample time steps ST_1 and ST_2 is calculated using these equations:

$$DM_i(ST_1, ST_2) = \sum_{t_1 \in ST_1, t_2 \in ST_2} \sum_{v} dif_i(t_1, t_2, v) \tag{7}$$

$$dif_i(t_1, t_2, v) = |\overrightarrow{f_i(t_1, v)} - \overrightarrow{f_i(t_2, v)}| \cdot (num(t_1, v) + num(t_2, v)) \tag{8}$$

where $num(t_p, v)$ is the number of feature points correspond to virtual word v at time t_p and $||$ represents the L2 norm.

For the location component, we calculate the average location of each virtual word from a user specified component of feature descriptors for each sample time step. An important component according to the data features under exploration can be selected for this purpose. The Equation 8 can be modified as follows:

$$dif_i(t_1, t_2, v) = |\overrightarrow{L(t_1, v)} - \overrightarrow{L(t_2, v)}| \cdot (num(t_1, v) + num(t_2, v)) \tag{9}$$

where $\overrightarrow{L(t_p, v)}$ is the average location of virtual word v at time t_p. The rest is the same as the applicable components.

To ensure the correctness of time lines for visualizing time-varying datasets, we need to handle the problem that different sample time windows may contain different numbers of time steps. If we simply follow the original Lowbow algorithm, the histograms for the sample time steps containing more feature points may collect information from smaller time durations, yielding an unequal property of the time lines. Therefore, we make the following changes to ensure the equal window size everywhere. We can first calculate a histogram for each time step and normalize them using the numbers of feature points. Then, we operate sampling windows on time steps instead of feature points. All the applicable components of feature descriptors can be treated in this way. Similarly, the average location should be calculated for each time step, instead of every sampling window. This process results in an effect that sampled time steps may have different numbers of feature points, meaning that feature points from different time steps have different weights on the resulting time curves. We believe that this is a necessary change to keep consistent local statistics of data distributions for time-varying data visualization.

4.4 Time Line Visualization

This section describes several methods to visualize time lines for different exploration and analysis purposes. First, we present a method to concatenate feature descriptors with the correspondence of feature points, which allows users to apply their existing knowledge or hypothesis of the data in the time line generation process. Second, an approach of parallel time lines is provided for simultaneous visualization of related information. Meanwhile, we describe a hybrid time line generation approach, which mixes input datasets, for comparing data from different time duration or attributes. Fourth, a focus+context method is employed to allow users to adjust the details of different time durations flexibly.

4.4.1 Concatenation of Feature Descriptors

Up to now, we generate time lines based on the distribution of virtual words. It is also very useful to introduce the relationships among virtual words, as sometimes a small set of correlated features, such as the control points to model a human face [161], can describe a dataset very well. For time-varying data visualization, we allow users to identify such correlations among virtual words and specify a sequence to concatenate them during the generation process of time lines. Users can apply their existing knowledge of the data to generate corresponding time lines or explore hidden temporal relationships through hypothesizing different sequences.

Specifically, we allow users to specify a sequence for a set of virtual words and use it to generate new, long feature descriptors by concatenating the same applicable components of feature descriptors. Instead of using distribution of virtual words, we collect histograms of concatenated virtual words for sampled time windows. The rest of the procedure is the same as previous. We cluster long feature descriptors and generate histograms to be used in Lowbow. This method ensures that all the feature descriptors have the same length for the clustering step. The sequence of virtual words can be determined by important locations in the space of data properties. In analogy to the control points for human face modeling, each virtual word can be related to a special meaning of the datasets.

Figure 15 shows an example time line for the attribute NH3 of a January air quality

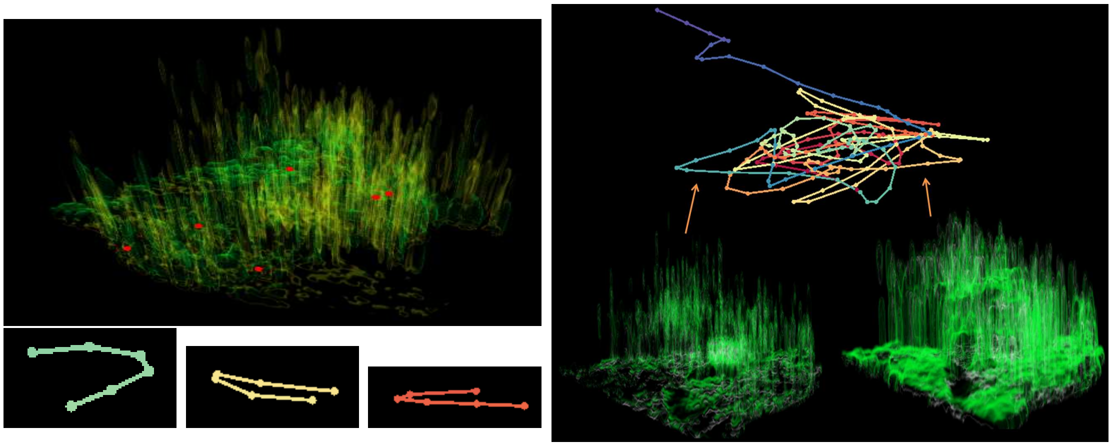


Figure 15: An example of concatenating feature descriptors. The image on the left top shows an example volume rendering of one time step. The red points indicate the locations of virtual words, which correspond to several regions with the maximum data variations in the entire time range. The times lines on the left bottom are time lines of three days as examples, all suggest that the time steps at the start (early morning) and the end (late night) are similar. The time line on the right visualizes this change across a month, with pollution levels high in the day (right snapshot) and low at night (left snapshot).

dataset. For this dataset, we know that several regions during this time duration vary the most. Therefore, we select a few regions as the key portions of the data, as shown on the top of Figure 15. For each region, we choose the feature point that is closest to the region center as a virtual word. The long feature vectors are then generated by concatenating the feature descriptors of each virtual word. In the time line, each day is represented with 6 sample time windows. The majority blue to red portion of the time line suggests that data changes repeatedly from left to right, which correspond to the major air quality distribution feature - high pollution levels during daytime and low levels at night.

4.4.2 Parallel Time Lines

We explore parallel time lines to visualize the relationships between different data properties. Parallel time lines are generated by shifting the original time line along a certain direction on the 2D plane. This provides a visualization of multiple data properties simultaneously. It can be used to analyze the relationships between several data properties or different data fields across time.

We first determine the shifting direction to reduce the overlapping issue of parallel time lines. With the locations of sample time steps, we apply the principal component analysis (PCA) algorithm [149] to calculate the main directions of point distribution. The shifting direction is identified as the second (less important) direction of the PCA result. Second, we allow users to adjust the amount of shifts to achieve their ideal effect. Since the rendering parameters of a time line are very limited, we also explore controlling parameters of time line visualizations. For effective comparison, we need to limit the number of parallel time lines and the choices of different rendering parameters. For instance, the number of the used color for visualizing different time lines can not be too large to ensure that the result is easy to recognize. Figure 16 shows an example of parallel time lines for two types of information simultaneously. It is obvious to see the time durations when these data values are different.

4.4.3 Hybrid Time Lines for Data Comparison

To visualize the relationships between different time durations or data fields, we can also mix their datasets in the generation process of time lines. In this way, sample time steps from different data will be distributed on the same 2D plane according

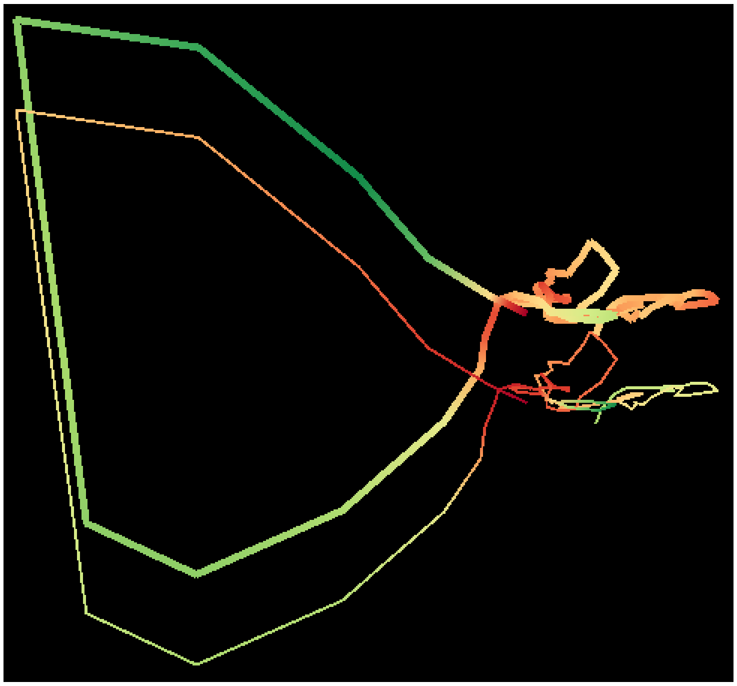
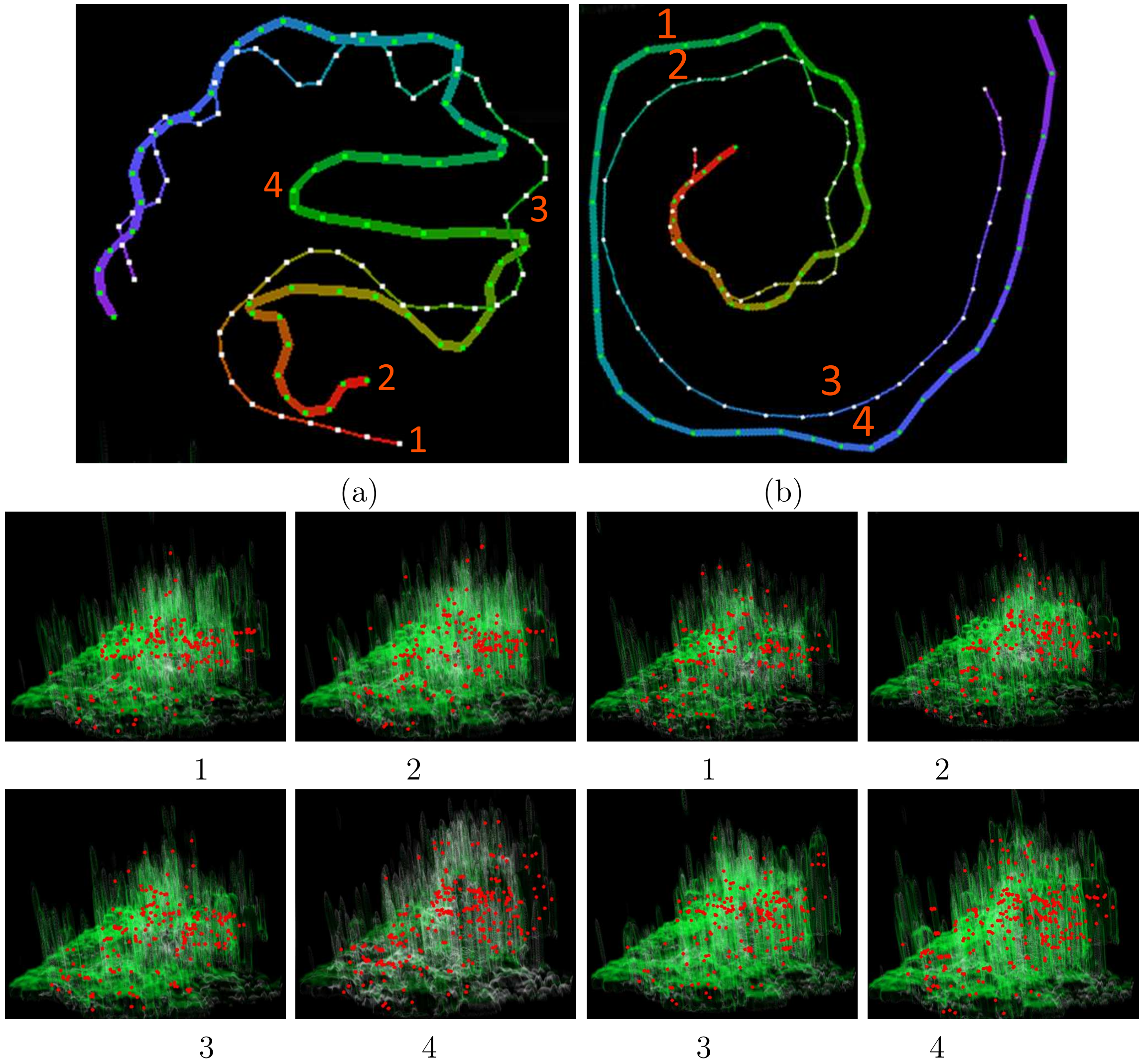


Figure 16: We use parallel time lines to visualize the amount of feature points (the wider line) and average density (the thinner line) for an energy dataset. The color represents values from different sample time steps.

to their dissimilarities calculated from Lowbow. This allows users to compare their relationships across time closely. To distinguish different time durations or data fields, we generate separate time lines and render them with different parameters. Figure 17 compares two pairs of different time durations (January 11-21 and 16-26) of attribute NH3 in an air quality datasets. We use different line widths to differentiate the two time durations. The time line from (a) shows that the two time durations are similar, and the time line from (b) indicates that the two green portions in the middle are significantly different.

4.4.4 Multi-Scale Visualization

A multi-scale visualization is provided to users for adjusting the details of different time durations. We separate the visualization process into two independent steps:



(a) (b)

1 2 1 2

3 4 3 4

Figure 17: With the time lines, we can easily compare different datasets or time durations. The numbered snapshots from separate volume renderings below the time lines provide details of the selected time steps.

determining sample time windows and generating time lines.

In the first step, we initialize the importance values with 1 for all the time durations. Then, we update it according to user selections. User can select the amount of sample time steps and adjust the importance values of different time durations respectively. With this information, we compute the number of sample time steps for each time duration, yielding an even distribution of importance of duration t as follows:

$$Avg_{im}(t) = n(t)/(imp(t) \times length(t)) \tag{10}$$

where n(t) is the number of sample time steps, imp(t) is the importance value, and length(t) is the length of the time duration t. We then distribute sample time steps evenly within each duration. The more sample points the time duration have, the more details will be revealed within the time range.

To smooth a time line, adjacent time windows are overlapped. Generally we use five times the distance between two sample time steps as the size of time window. This parameter can also be interactively adjusted. The second step is to generate a time line with a histogram from each sample window.

Figure 18 shows the time lines of two synthetical cube datasets with different sizes of sampling windows. Figure 19 shows three time lines from user interaction. The time line (a) enlarges the time duration in the middle around the extrema location in orange. The time line (b) enlarges the portion in blue to yellow to observe the subtle details of the normal periodic trend. The time line (c) demonstrates that our approach allows flexible selection of important durations. By increasing the number of sample time windows, we can see more details in the time lines, reflecting more

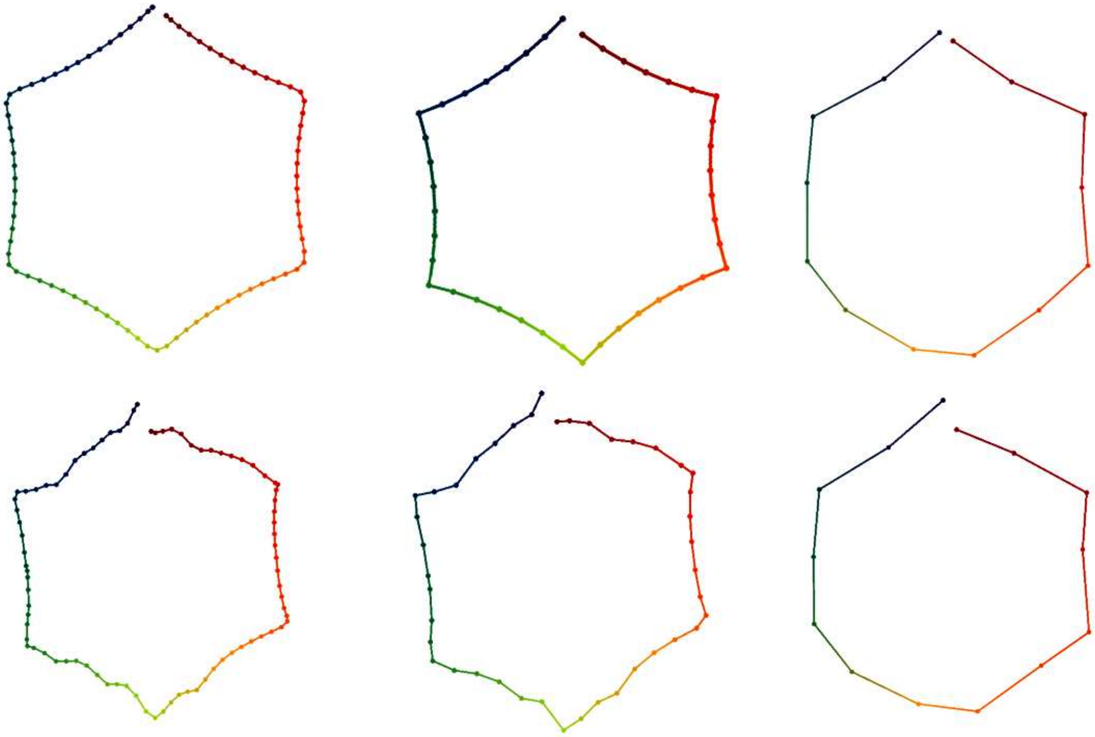


Figure 18: Results of the multi-scale visualization from two Cube synthetical datasets. The first row shows the time lines for the cube moving around dataset, illustrated in Figure 14 (a). The second row shows the time lines for the cube under noises dataset, illustrated in Figure 14 (e). Different sample time steps (84, 42, and 14) are applied to the three columns respectively. This figure demonstrates that the statistics aspect of Lowbow can reduce the effect of noises in the time lines.

accurate information of the temporal durations.

4.5 Results

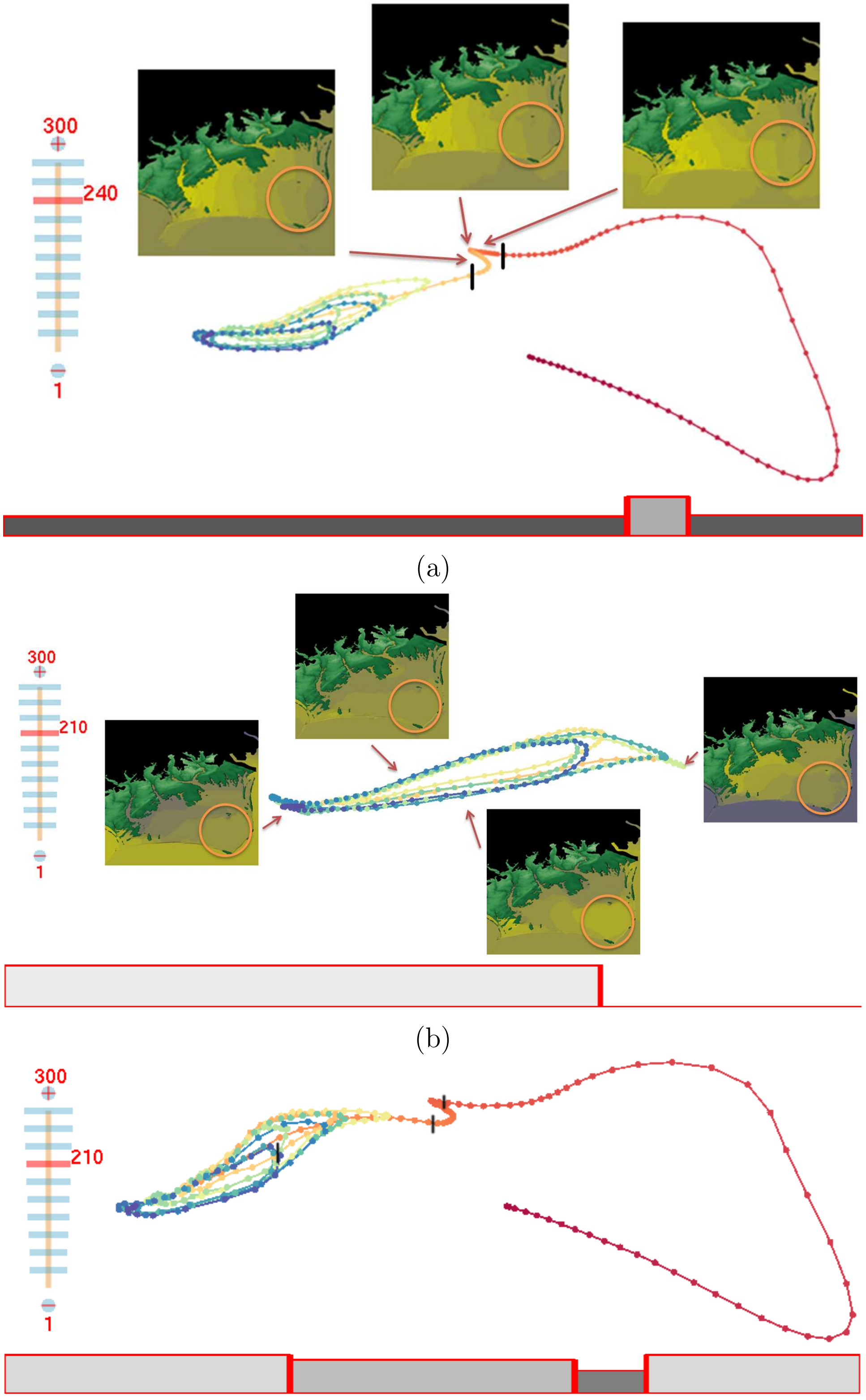
Our approach produces the visualizations of a single or multiple time-lines. The snapshots and arrows are added manually in Figures 15, 17, 19, 20, 21, and 22 for illustrating selected time steps. Any volume rendering methods can accompany our approach to visualize individual time steps. In the following case studies, we apply time lines to study the major temporal relationships in the time-varying datasets.

4.5.1 Case Studies

4.5.1.1 Test Case

To demonstrate the effects of time lines, we concatenate three different datasets, air quality, flow tube, and energy, to compose a time-varying dataset. As shown in Figure 20, the automatically generated time line clearly shows three time segments with different color. In this test, the differences between the three datasets dominate the shape of time line, resulting in three clusters on the time line. When we focus on the main temporal change by using a small number of sample time steps, shown in Figure 20 (b), we can identify the three time segments easily. Larger numbers of sample time steps can be used for revealing more details of each time duration, as shown in (a) and (c).

We start the data exploration with uniform sample time windows, which means that the locations of sample points between two datasets are affected by both of them. This explains why the three line segments are connected. If users are interested, they can further select one of the time duration, e.g., Figure 16 only visualizes the duration in



(a)

(b)

(c)

Figure 19: Multi-scale visualization results of the storm surge dataset illustrated in Figure 21. The interaction panel at the bottom shows that users can adjust time durations and importance degrees flexibly. The scale bar on the left indicates the number of sample windows for the current time lines. Snapshots from user selected time steps are also provided.

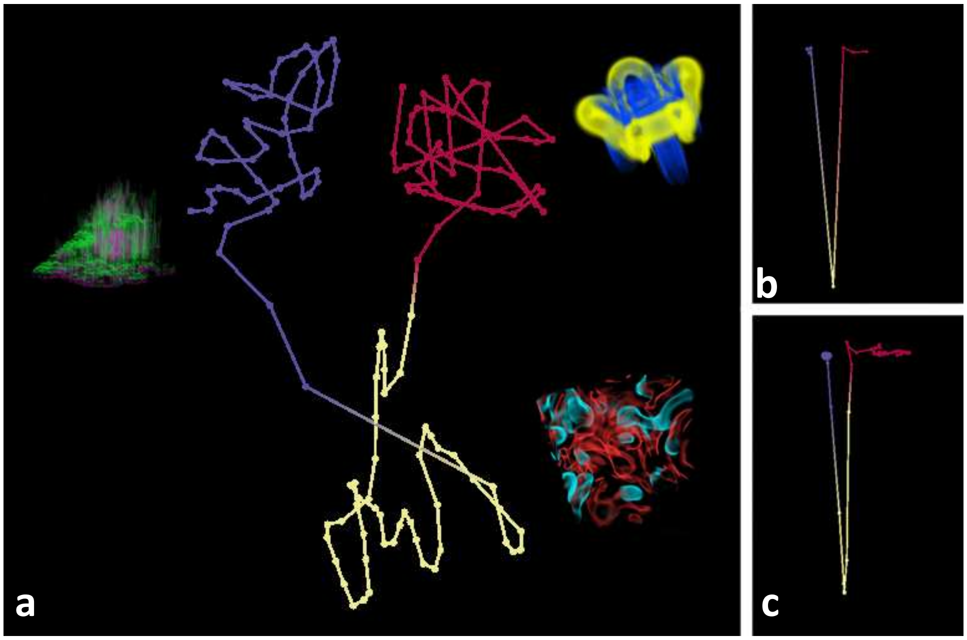


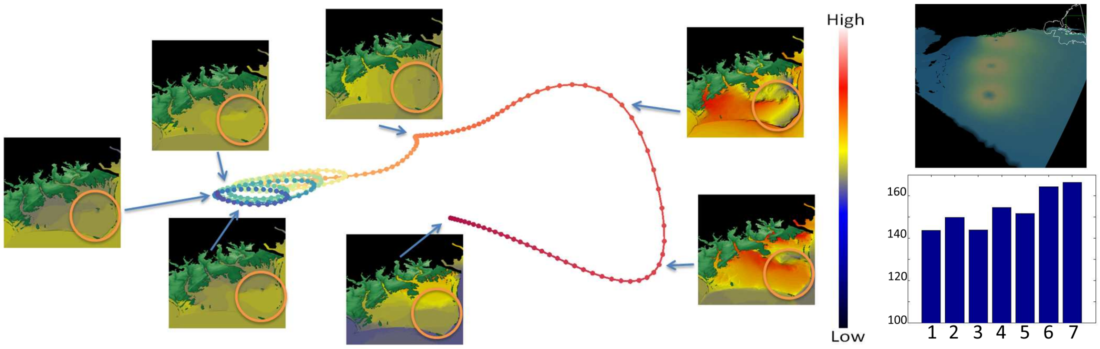
Figure 20: Three time durations can be identified easily with the time line visualization. We generate three time lines by choosing different parameters (numbers of sample time steps, window sizes) for exploring different details: (a) (150, 10), (b) (60, 30), (c) (90, 10).

purple, and we can analyze more details from there. This test shows that time line visualization helps users to understand that there are three main temporal events, which segment the time duration easily.

The following provides two case studies for a time-varying storm surge dataset and a multi-field time-varying air quality dataset.

4.5.2 Storm surge data visualization

We apply the time line approach to study the storm surge data simulation of hurricane Isabel. The dataset we use in this case study contains information around the sea shore at North Carolina. All the snapshots in Figure 21 are colored in the same way: green is used to render terrain and the rest color of blue to red represent the heights of ocean.



(a) (b)

Figure 21: (a) The time line of a storm surge dataset demonstrates the effects of Hurricane Isabel to Outer Banks, North Carolina. The left portion (blue to yellow) corresponds to the periodicity of tidal elevations that influence every 12.4 hours. The right portion (orange to red) indicates significant data changes of ocean surface from its normal distribution during the hurricane. The circle in the snapshots indicates the user specified regions. (b) Top: Hurricane path is shown with three example time steps. (b) Bottom: The average heights of ocean levels are increasing before the hurricane, which confirms the shifts on the time line.

As shown in Figure 21 (a), the time line visualizes several important temporal trends, which are difficult to detect otherwise. First, we can detect a normal periodic feature (blue to yellow) and the effect of hurricane (orange to red), which corresponds to the abnormal high ocean levels in red as seen from the two snapshots on the right. Second, the time line suggests the starting time step of hurricane automatically with the extrema curve location in orange. Around the starting point, we can see a waiting period before the abrupt of hurricane according to the dense point distribution. Third, the distance between adjacent sample time steps increases, which indicates that the data changes dramatically during hurricane. Fourth, the linear curve shape of the hurricane duration is consist with the movement of hurricane center roughly in a line, shown in Figure 21 (b) Top. This movement event is similar to the cube examples in Figure 14 (a), which can be identified from the line shapes as well. Fifth, the direction of the red portion at the end indicates that the data changes back toward its normal distribution after hurricane, as it moves toward the periodic cycles.

We can continue to explore the details of temporal trend by enlarging selected time durations, such as the periodic cycles shown in Figure 19 (b) Top. As the cycles move toward the hurricane portion, we can expect that the ocean levels are increasing gradually, which is confirmed with the average height per cycle measured from the data, shown in Figure 21 (b) Bottom.

4.5.2.1 Air quality data visualization

We also apply time line in the study of air quality with multi-field time-varying datasets. We use a continuous air quality simulation from the CMAQ model in 12 months, of which each location takes 25 attributes. All attributes are collected 25 times every day.

We first generate time lines of four attributes respectively to visualize the data changes in a year. The number of sample time steps for this figure is 365 and the window size is 50. The top four time lines in Figure 22 are colored from blue to red, representing Jan to Dec. Each time line visualizes an important data property: pollution volumes in the early spring and late winter are similar (the blue the red segments), and are different in the summer and fall (the green to yellow portion). This property is confirmed by domain scientists. Also, simply taking the time lines generated independently, we can detect that attributes NO and NO2 are similar, while NH3 is very different. This finding is consistent with the CMAQ model, since NO2 is generated according to the amount of NO.

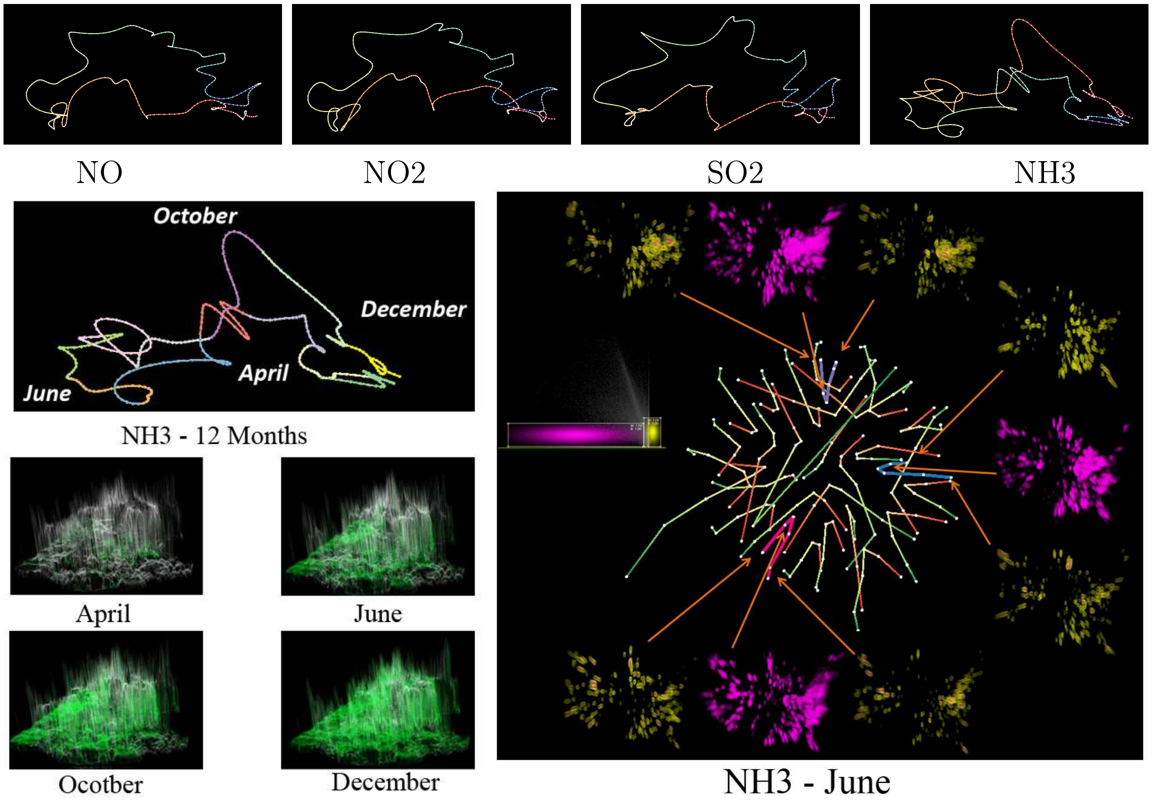


Figure 22: (NO, NO2, SO2, NH3) Time lines for four attributes of an air quality dataset for a whole year suggests that NH3 is very different from the other three attributes. (NH3 - 12 months) To further inspect attribute NH3, we use 12 different color to represent 12 months in one year. (NH3 - June) This figure shows time line of attribute NH3 in June, colored by the number of days. The snapshots show the morning, noon and night renderings of three individual days, which are colored in pink, blue and purple. With every day starting at green and ending at red, obvious patterns can be observed.

Further, we focus on the special attribute NH3 and apply different color to represent the 12 months in a year. As shown in Figure 22 (NH3 - 12 months), we generate the average 3D data visualization to explore the temporal change according to the time line. The volume visualizations of four months provide visual contexts to understand that the pollution volumes change from the white and green bodies back and forth in a year.

We can further zoom into a smaller time range to study detailed changes. For example, we explore the attribute NH3 in June with a time line shown in Figure 22 (NH3 - June). In this time line, we increase the number of sample time steps to 180 and window size to 25, so that daily-level temporal evolutions can be revealed. We also modify time line visualization by omitting the line segments between adjacent days, thus only the sample time steps belonging to the same days are connected. This helps users to study the temporal changes across different days, which are shown with the line structure patterns. Generally each line starts from outside, reach into the center at day, and drop back at night. Such line patterns indicate that the pollution volumes in the early morning and at late night of a day are similar; the volumes during day time in this month are similar. This finding is further confirmed with the 3D visualization, shown on the outside of the time line. This visualization reveals that the data variations of all the days in this month are similar. Therefore, users do not need to visualize all the time steps for every day in this dataset. This demonstrates that the data distribution in a time line can significantly shorten the exploration process for users.

4.5.3 Quantitative Results

Table 1: The information of datasets and performances of key steps.

Dataset	Dimension	Number of Time Steps	Feature Extraction Time Sseconds/Time Step
energy	128^3	200	75
flow tube	100^3	249	50
storm surge	600^2	792	20
air quality	136^3	775	90
Dataset	Description Collection Hours	Clustering Minutes	Time Line Generation Seconds
energy	2	6	3 (100 windows)
flow tube	2	6	3 (100 windows)
storm surge	3	18	8 (150 windows)
air quality	14.8	80	20 (300 windows)

The performance of our algorithm varies from the data size and complexity. First, the selection of feature points depends on the data size and the number of time steps. Second, the calculation of feature descriptors is linear to the number of feature points. All the feature components are very fast to compute, except the SIFT component. Third, the extraction of virtual words basically depends on the sizes of selected feature descriptors and the number of feature points. Fourth, the generation of time lines is linear to the number of time steps.

Overall, the performance and memory requirements of our approach are linear to the number of time steps. The most time-consuming stages can be pre-processed. The first step is performed once for the entire dataset, leaving the interactive selection process much faster. The second and third steps are performed only once for a dataset and do not affect the data exploration process. The main algorithm used during the visualization process is the generation of time lines, which is less than a minute for the entire time-varying dataset. The exploration and visualization is interactive.

Solutions to accelerate the pre-process stage include using improved SIFT algorithms such as SURF [18], reducing the number of virtual words, and selecting regions-of-interests. Both these methods can achieve fast and accurate feature extraction and can greatly reduce the processing time. Also, since the computations of feature points and histograms of time windows are independent, we can employ GPU-based or parallel computing techniques to accelerate the pre-process state. These methods are especially used during exploration stages to help users to identify the regions-of-interest or time durations.

4.6 Discussion

4.6.1 Discussion on Visual Design

The efficiency of the time line visualization comes from the visualization of data relationships with 2D distances. Cognitive science has shown that human beings can effectively recognize object similarities from a representation by their distances [23]. Since the time lines distribute sample time steps on the 2D space according to their data features, users can easily identify some interesting events, such as periodicity, similarity, and important temporal events through extrema locations.

The major advantage of time line is its capability to incorporate the entire time-varying dataset into one single line to show its progression along the time. The modified lowbow algorithms allows user to show the temporal changes of dataset at different scales. For example, the user can display the time line of the air quality data at a monthly, daily or even hourly basis. This ensures that time line can handle long time sequence of data. Other approaches, which is based only on sampling through the entire dataset, will lead to information loss and inaccurate results.

Though our time line approach shows only the arbitrary shapes and does not have details as important regions inside each volume, the line is generated based on the features extracted from the dataset. It is easy for users to identify abnormalities and further explore the dataset to find events and features. It is impossible to visualize the dataset without any kind of data reduction when dealing with large-scale datasets. Without this step of information reduction, it will be more time-consuming to generate time lines. Note that, using feature points can still capture a wide range of data changes, particularly because we select feature points for each time step independently. When objects change significantly from previous time steps, new feature points often occur. Successful cases, like face recognition, have already shown that a few feature points can help reduce a large amount of irrelevant information.

Our case studies demonstrate that time lines are a very useful addition to rendering results for data exploration. Visualizing data corresponding to locations on a time line can help users understand the distribution and relationships of different time steps. With a time line, users can visualize just several time steps to obtain a quick understanding of the entire time-varying dataset. Our approach is more efficient than snapshots or animation, since it does not require rendering results from all the time steps. This is especially important for analyzing datasets with a large number of time steps. On the other hand, we realize that it may take time for users to get familiar with time lines. Our results of the cube dataset can be used as an example.

We think that interactivity should be relevant to the data size. For ultra-scale datasets, only the I/O performance will be below interactive speeds. The processes of selecting feature points, collecting feature descriptors, and clustering of virtual words are time-consuming. These processes should be performed ahead of the data exploration. During visualization, users can still adjust their focuses on the time lines and visualize different time durations and data attributes with a reasonable performance. For large scale time-varying datasets which include millions of time steps, the time line approach can save the exploration and visualization time for users significantly.

Generally, time lines are useful to visualize significant data changes. Small details may be lost among the overall temporal trend. Our approach allows users to choose transfer functions, select descriptor components, adjust the amount of feature points, and fine-tune level-of-details. These interactions can assist users to explore a complex dataset gradually.

4.6.2 Comparison

We compare our approach with four different methods. Among which, time histogram and time activity curves are popular approaches for time-varying data visualization. The last method is a different design based on our extraction method of virtual words.

4.6.2.1 Time Histogram

Time histogram is a popular method to visualize the distribution of data properties across time. Generally, the horizontal axis represents time and the vertical axis represents data properties, such as data values.

The left image in Fig 23 shows time histogram generated based on density values with 100 bins. The right one is generated using the extracted virtual words. Each column represents one sample time step and each bin represents one virtual word. There are 100 sample time steps and 500 words in the vocabulary.

Advantages of Time Histogram: Time histogram is simple to compute and straightforward for users to understand. It has also been applied widely in many fields. As shown in Figure 23, the density histogram on the left indicates that the flow becomes stable during the second half of the time range. This is a great method when the data size is small and important data features are related to the voxel values directly.

Advantages of Time Line: Compared to time histogram, time line can visualize general temporal trends; while time histogram may miss many important temporal trends, as it only suggests the similarity of value distributions. Time line allows selection of data features beyond voxel values, which is crucial for exploring datasets with complex features. For datasets with long durations, time line can statistically sample the dataset at different scales; thereby producing visualization in a suitable resolution.

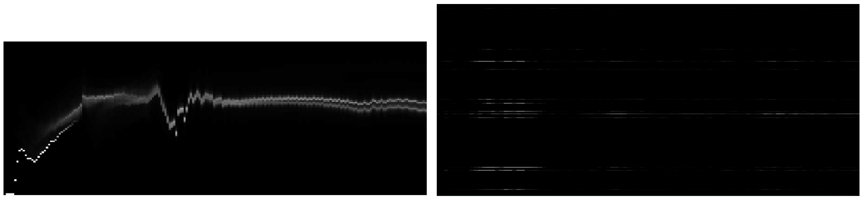


Figure 23: Left: Time histogram of an energy dataset; Right: Virtual word histogram of energy dataset.

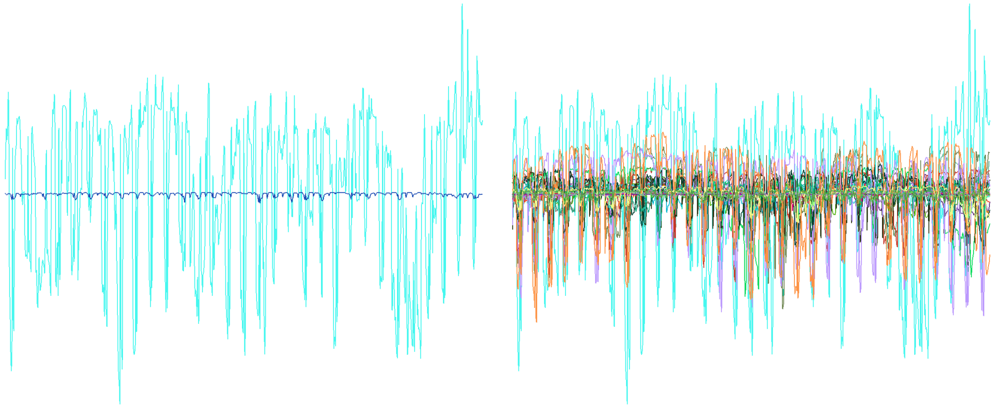


Figure 24: Example time activity curves of several blocks of voles from air quality dataset. Left: only two curves are shown. Right: one hundred curves are shown.

4.6.2.2 Time Activity Curves

Time curves are first used to reveal the temporal changes of time-varying medical visualization. Given a spatial location inside a body, the amount of activity measured is correlated to the biological function which is imaged. Many forms of time curves are then proposed and applied to scientific and volumetric datasets [51, 151, 141]. We have briefly described these approaches in the section of related work.

As shown in Fig 24, the curves are generated based on L1 metric from Equation (1) in [51]. One hundred data blocks are used to render the time activity curves.

Advantages of Time Activity Curves: Time activity curve integrates the advantage of time histogram, which is intuitive to understand; and advantage of time line, which allows selection of various data features. Time activity curves work well for some applications, such as identifying functions of organs [51]. Since the location of organ is known before visualization, all the voxels within that spatial location can be clustered and calculated into one curve to represent the biological function.

Advantages of Time Line: First, time line does not have the clustering issue of time activity curves, which produce a number of curves for data blocks. Also, time line visualizes temporal trends directly; while time activity curves, the same as time histogram, require users to analyze temporal trends through similarity of curve values. Further more, time line represents similarity of two time steps with distances on a 2D plan, which does not require users with professional trainings.

4.6.2.3 Storyboard

The closest previous work to this approach is the interactive storyboard approach [94] for visualizing overall data contents and relations of time-varying datasets.

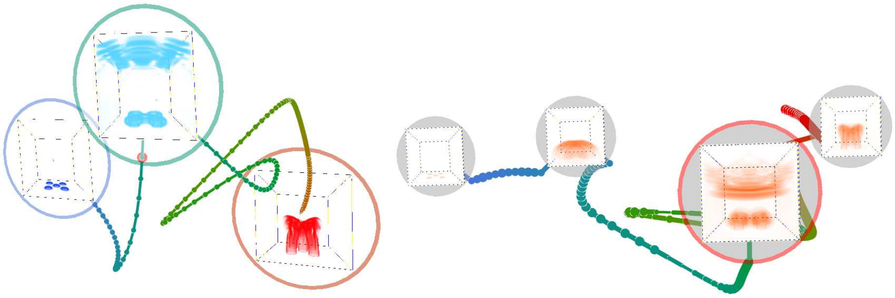


Figure 25: Example visualizations of the energy data at two scales from the storyboard approach[94] .

Advantages of Storyboard: As shown in Figure 25, the storyboard approach selects representative time steps from a time line and embeds their snapshots in the storyboard visualization. This design is convenient for users to understand the overall data contents and changes through both the example snapshots and changes of time lines.

Advantages of Time Line: First, the dissimilarity comparison of storyboard is based on data blocks. Every block in a time step is compared with the data block at the same location from all the other time steps, leading to n^2 (n is the number of time steps) performance. While time line can independently collect distribution of virtual words for each time window and compare distributions of each two time windows, leading to $n + h^2$ (h is the number of virtual words) performance. Since h is much smaller than n ($n \in [5, 800]$ is generally much smaller than n), time line is more efficient than the storyboard approach. Second, the feature definition of storyboard approach only includes simple measurements such as data density, gradient and second order statistics. Time line allows feature selections based on representative voxel features/virtual words.

4.6.2.4 Visualization of Time Window

As seen in Fig 26, bags of local virtual words are shown in histograms, which have time axis vertical and number of words horizontal. The histograms show how each small time duration are different from others. The brightness of each row reflects the amount of the particular virtual word inside the local time window. We can observe the differences between these two time stages.

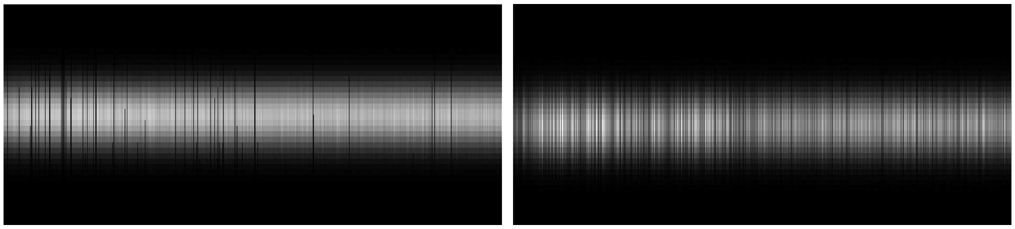


Figure 26: Time window visualization of a hurricane dataset at early (left) and late (right) stages.

Advantages of Time Window: The time window clearly shows how many different words are with in this time duration, which promotes the analysis of individual data features.

Advantages of Time Line: Time line is scalable to the number of time steps. Compared with time window, time line can handle large datasets with its statistical sampling component.

4.6.3 Discussion of Choices

In our approach, we project the temporal differences to a 2D space with MDS; therefore, the time line is an approximation of the original difference measurement. This is unavoidable when representing a high dimensional data in the 2D space. Indicated by various studies that apply MDS [130, 24], the MDS-based dimensional reduction scheme produces an optimal approximation of the temporal differences. Our approach, includes the measurement of temporal differences and later data visualization and comparison, can be integrated with other dimension reduction methods, such as PCA (Principle Component Analysis) [80], LLE (Local Linear Embedding) [113], LDA (Linear Discriminant Analysis) [55, 58]. We choose MDS because of two reasons: first it is widely used in information visualization and graph visualization; second it is based on a matrix of similarities and assigns a location to each item in N-dimensional space, which matches our requirements exactly.

Some of steps in our approach can be simplified or replaced with other algorithms. For instance, we can use other feature extraction methods such as Harris Corner detection [69], SURF [18], or simple dissimilarity measurements like voxel values. All these methods are suitable for some types of data features. In particular, SIFT algorithm can capture gradient-based features and object boundaries very well, which are often important features for scientific datasets. We can adjust the parameters to concentrate on different types of features: for example, reducing the number of gradients in SIFT descriptor can point to features related to data values. The choice of the feature extraction methods should be finally determined according to the data properties and domain knowledge.

4.7 Conclusions

In this chapter, we present an approach to generate succinct time lines for visualizing time-varying datasets. This approach is derived from the fact that human languages can describe various temporal events effectively with a limited set of words. For time-varying datasets, we characterize a set of virtual words according to selected data features and treat the entire dataset as discrete descriptions with these words across time. In this way, many important temporal relationships can be detected through measuring the distribution of virtual words. We further summarize temporal relationships as time lines by projecting their dissimilarities to the 2D space. With provided interaction methods, our approach assists users to explore and analyze a time-varying dataset with a simple, yet effective tool.

CHAPTER 5: ENSEMBLE VISUALIZATION

5.1 Introduction

Scientific simulations are generated to study various natural phenomena and behaviors. These simulations are often produced on super computers with different input parameters. With the techniques advancing, the resulting data sets are growing both in size and duration. They are usually large, multi-variate, multi-dimensional, and spatio-temporal. By carefully studying the simulations, scientists can gain an understanding of possible outcomes of each simulation parameter. An ensemble data set is a collection of these simulation.

Normally a large number of parameters are involved in the process of generating a simulation. Take tropical cyclone as an example, scientists and emergency responders are extremely interested in how and when events, such as inundation, hurricane movements and elevation, happen. Predictions are made through the researching on such ensemble datasets. This is important for emergency responders to make decisions such as evacuation. In this case, better visualization systems are required to provide assistance to summarize ensembles, analyze correlations between variables and members, study underlying features.

The main challenge to visualize an ensemble data set is its complexity and size. Moreover, due to the large number of ensemble members, any visualization technique will be too visually cluttered to provide assistance in data analysis. In this chapter we introduce an ensemble visualization approach which visualize the ensemble and allow user to perform exploration and analysis within one ensemble member or among members of different parameters.

Our approach consists of two parts. First, we utilize the time line approach we described in Chapter 4 to generate lines to represent a large number of ensemble members together. We analyze the distribution of local statistics and classify members into groups. We highlight each group and hide the members in our visualization to provide visual summary of the ensembles groups. Second, we propose a method to automatically add newly generated simulation members into the visualization without comparing it with all existing members. We achieve this by comparing the new member's local statistics histogram with the existing members'. This does not require additional calculation of the dissimilarities through MDS. In this way, the speed of rendering such a large simulation datasets can be greatly improved.

The related work of ensemble visualization is reviewed in Section 2.4. The rest of the chapter is organized as follows. Section 5.2 describes our detailed approach and results. Finally, Section 5.5 discusses our approach and concludes this chapter.

5.2 Approach

5.2.1 Ensemble Datasets

As described in Section 3.2 the datasets, we mainly use two ensemble dataset: storm surge and oil spill. Each of them is scientific simulations with different input and output parameters over time and space. The storm surge dataset contains a large number of days with multiple variables on each day. The oil spill data has particle movement information for different simulation parameters. Both of them are ensemble dataset with large amount of information.

One property of the ensemble data is that the members share common input parameters, so their time line presentations are similar. But with the parameters changing, each of them may be shifted and slightly different from the previous ones. All of our datasets simulates scientific scenarios on a single day or a same time duration, so they all share some periodic patterns with each member's time line lying near others. As shown in Figure 27, one time line to visualize an ensemble without tropical cyclone is displayed on the left, and one ensemble member with tropical cyclone is on the right. Each ensemble member contains a time duration of three and half days. The time line of a single member without a cyclone shows more stable patterns, each day shares a similar location. The pattern just suggests periodic tidal waves at this time duration. The one with cyclone shows the periodic pattern of each day as the time line goes left and right, but each day is different and the effects of a cyclone is clearly observed.

5.2.2 Ensemble Time Lines

We use our time line approach to represent the overall comparison of the ensemble dataset. Each simulation run is rendered as one time line and different simulations are connected. The first step is to load in a portion of the ensemble dataset since it is impossible to read in entire data. For example, each storm surge simulation consists of more than half a million vertices and dozens of variables, several simulations will

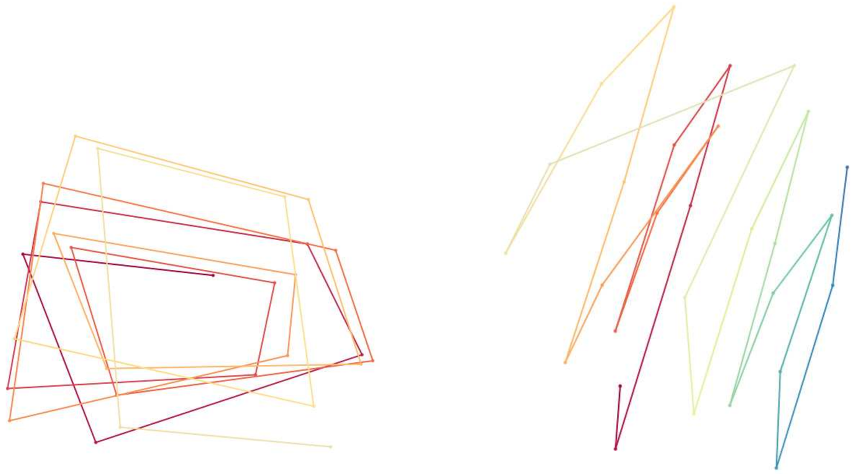
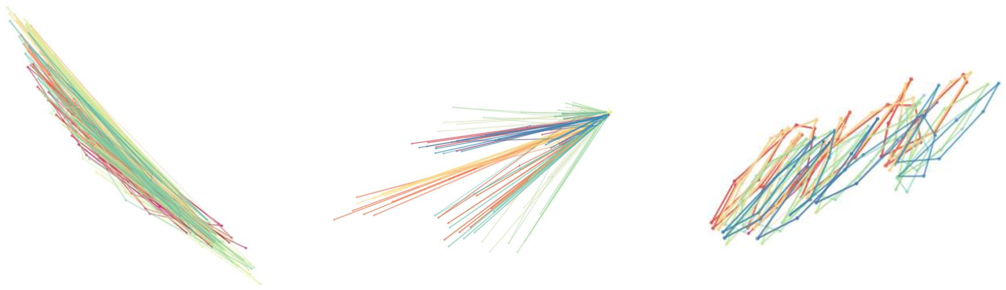


Figure 27: Left: time line of one ensemble member without tropical cyclone. Right: time line of one ensemble member with tropical cyclone.

take up all the desktop PCs memory. We select feature aspects and follow our time line generation approach by connecting all the simulation data together to create one single time line. Details of initial time line generation is covered in Chapter 4.

In the storm surge ensemble data, we connect daily simulation runs as a single long sequence and use sample windows to capture local features as before. Thus, the whole time line not only represents all the simulation data changes, but also still preserves individual simulation properties.

We test our approach on different ensemble datasets. As we can see in Figure 28 (a), an ensemble dataset of storm surge is visualized. The time line of the 40 connected simulations is shown in periodic patterns. Each day is rendered using different color. Figure 28 (b) shows 10 members from oil spill data. Each run is with different input parameters. The time line is created based on average moving speed of particles. Figure 28 (c) shows a month of air quality dataset. Blocks of volume are analyzed in this case.



(a) Storm Surge (b) Oil Spill (c) NH3

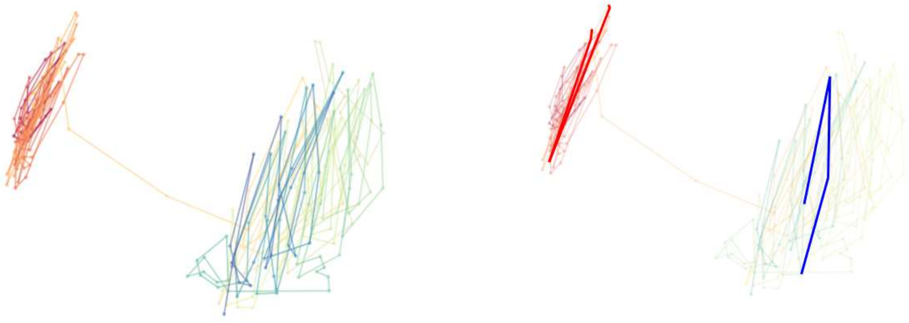
Figure 28: (a) Time line of 40 ensemble members from storm surge dataset. (b) Time line of 10 ensemble members from oil spill dataset. (c) Time line of 1 month data from air quality dataset. Each run is rendered with same transparency.

Figure 29 shows the ensemble time lines of the air quality dataset which only generate the time lines based on a small data block in each volume. In Figure 29 (a), we use about 40 days of simulation in two months of the air quality NH3 data blocks and for each day we use 5 sample points. The differences between months are huge in this simulation, the days in different months are clearly separated. Figure 29 (b) shows the rendering results of our visual cluster reduction method results described in Section 5.3. Two representative days are rendered to avoid visual cluttering of the entire dataset.

5.3 Visual Clutter Reduction

In our storm surge dataset, one major parameter changes during the storm surge, resulting in two very different groups of time lines as shown in Figure 30. Each of the group has similarities within their own group. Each group shares a similar periodic pattern.

One problem created by this approach is that with numerous simulation runs, there



(a)

(b)

Figure 29: (a) Time line of 40 days of air quality data, each member is sampled to 5 sample points. (b) Time line rendered by our visual clutter reduction method, using representative days to represent the whole group.

may be serious occlusion issues. When only some parameters of each simulation run changes, the output data can be quite similar with each other. So the resulting time lines have very close locations. Since each day is rendered using different color with a same transparency, they are visually cluttered and hard to observe any details when placing all the days onto a 2D plane.

To solve this issue, after we get our initial plot results of each individual day on the 2D plane, the plots of the first day will be used as a reference. We calculate the distance between the local histograms of reference and the next simulation. We use $H_{(i,n)}$ to represent the local histograms and H_{ref} to represent the histogram of the reference day. The following the equation is used to calculated the distance between simulation runs and reference.

$$Dis(H_{(i,n)}, H_{ref}) = \sum_{n} \overrightarrow{H_{(i,n)}} - \overrightarrow{H_{ref}}. \tag{11}$$

A threshold will be setted determine wether the daily simulations are close to the

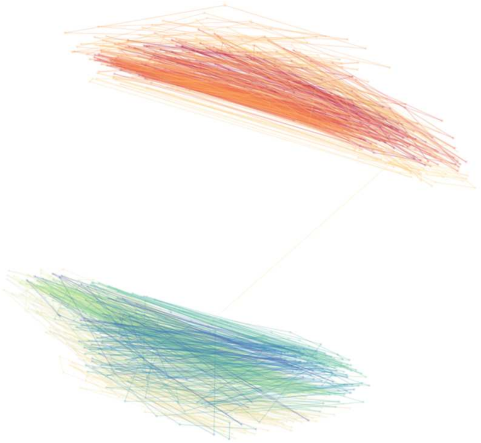


Figure 30: Time line of 40 ensemble members. The first member is opaque while other members are transplant to avoid visual cluttering. Time line of 41 ensemble members. The outlier is highlighted and form a new group.

reference. The days with a small distance to the reference will be rendered in transparent colors. If the simulation member is considered as similar to the reference, we update the reference using by calculating the average of the average histogram of simulation members in this group. When the distance between the simulation exceeds the threshold, another group will be created. A new reference day of this new group is generated too. The simulations coming into the ensemble later will be compared with all the existing references to decide if it belongs to the current groups or it needs to create a new group.

Figure 31 (left) shows the result of time lines in a same group with different transparency. The dominate periodic feature is preserved without visual cluttering. The first day in this group is highlighted to represent the whole group. Then another odd daily simulations comes in, which is very different from the reference, As in Figure 31 (right). We can clearly see this member with significantly different behavior and does

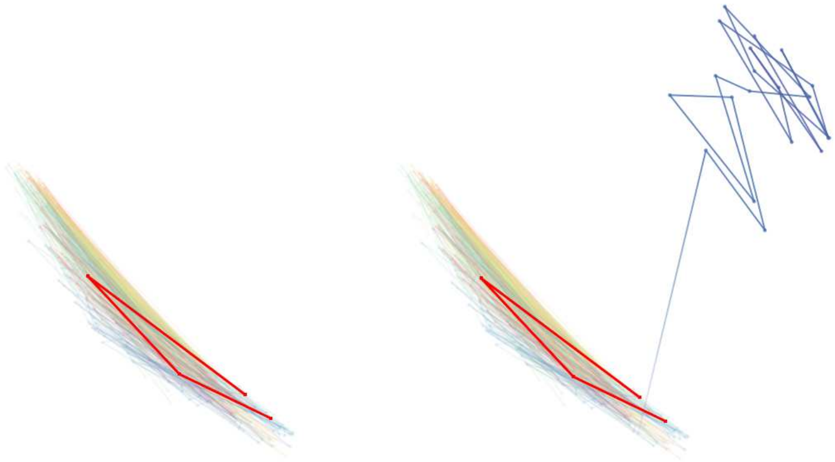


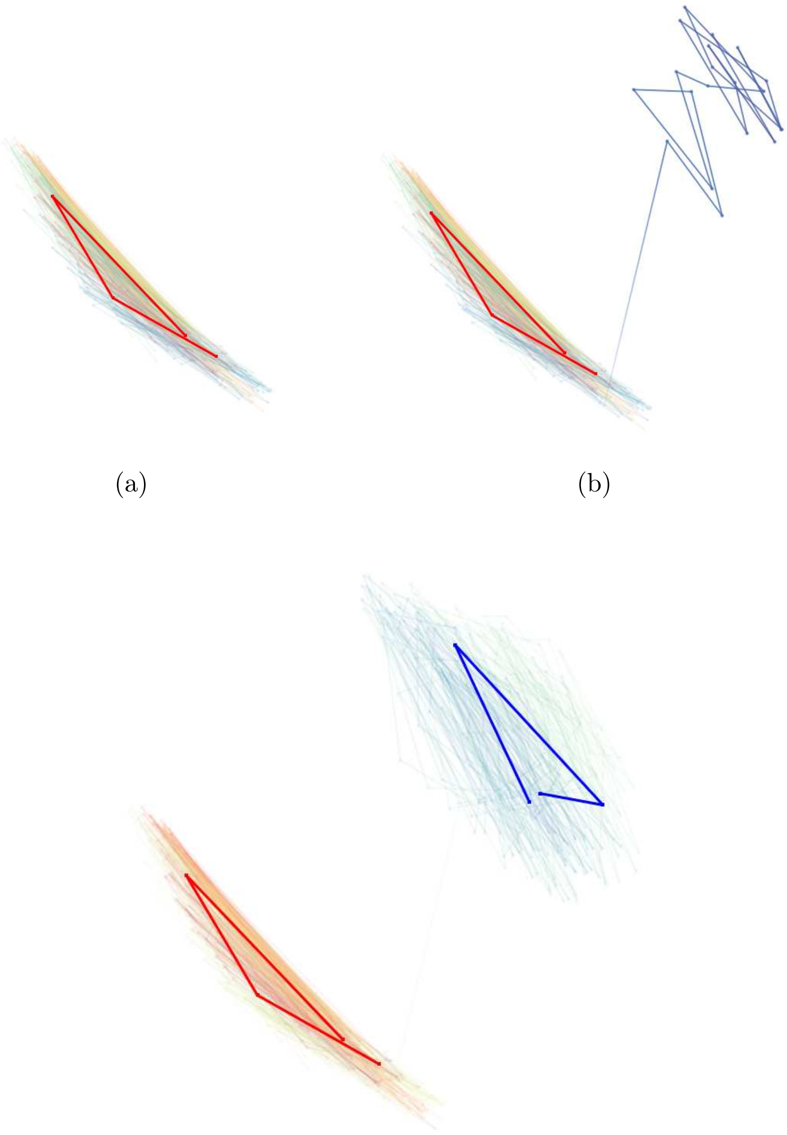
Figure 31: Time line of 40 ensemble members. The first day of the first ensemble member is opaque while other members are transplant to avoid visual cluttering. Time line of 41 ensemble members. The outlier is highlighted and form a new group.

not belong to the current group. The previous time lines are still in transparency but the new one is considered as an outlier and form its own group and rendered in full transparency.

We can employ a strategy to display the representative simulation run in each group. We sort all the distances between the reference day and each simulation member and use the member whose distance is closest to the reference. Figure 32 shows the results of using the closest day to represent the group.

5.4 Acceleration For New Member Insertion

Another challenging problem is when new simulation is generated, traditional method requires to calculate its relationship with all the existing members in the ensemble. In our approach, initial time line is only generate with with the already produced ensembles. When a new member comes, traditional methods require to calculate the distance of the new member with the entire ensemble first, then place



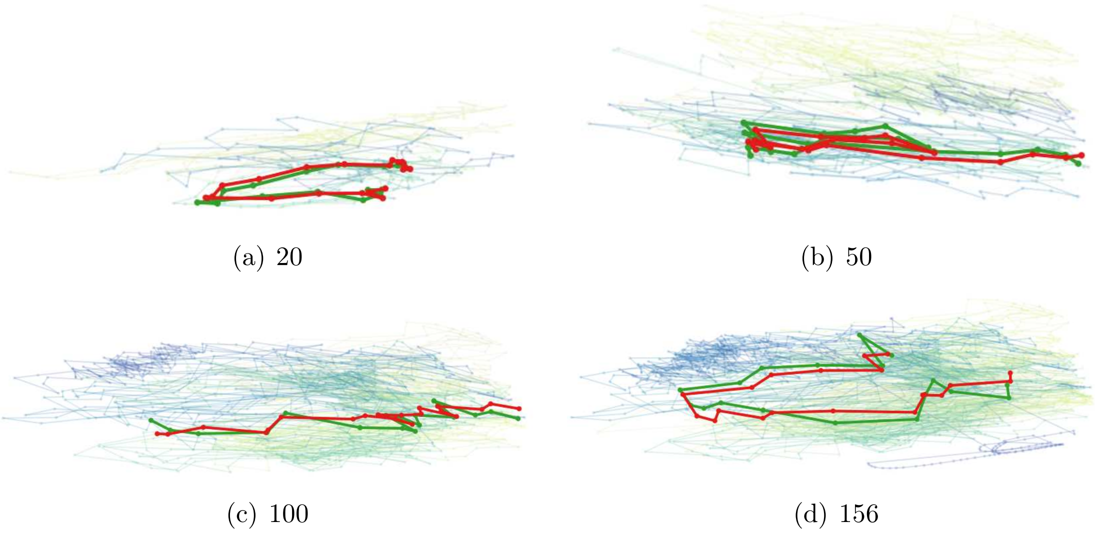
(a)

(b)

(c)

Figure 32: Time line of 40 ensemble members. (a) The first day of the member who is closest to the reference is highlighted to represent the whole group, while other members are transplant to avoid visual cluttering. (b) Time line of 40 ensemble members. The outlier is highlighted and form a new group. (c) Time line of 50 ensemble members. New members are considered as the second group and the first day of the member with the closest distances to the reference day is highlighted to represent the group.

it onto the 2D plane. This is too time consuming and unnecessary to use all the existing members again to calculate the whole layout. Besides MDS, we only examine the distance of the histograms of the new member to other histograms, and simply find the closest histograms to the new histogram.


(a) 20

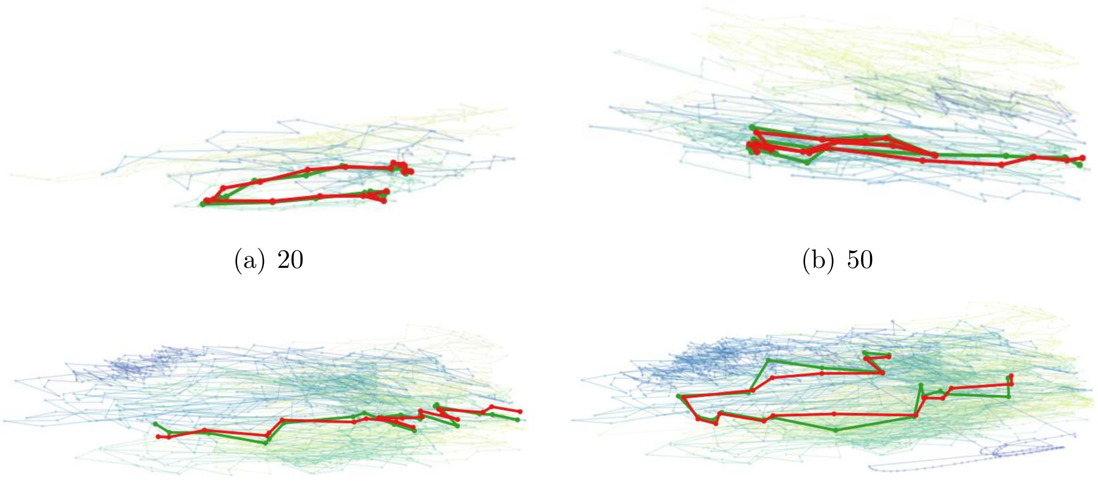
(b) 50

(c) 100

(d) 156

Figure 33: Sample result of inserting a new member into the ensemble visualization. The opaque time line in both figures is the new member. The green time line is our result and red is the pre-calculated position using MDS.

We test our approach using these three ensemble datasets. We first calculate the whole ensemble together and generate the time line, the last member is render in red. Then we use our method to find the suitable locations for the last member, our result is in green. This method could save a lot of time by not calculation the entire ensemble again. Figure 33 shows three examples of inserting a new member into the visualization. Each examples contains different sizes of ensemble members, Figure 33 (a) shows a small ensemble data with 20 simulation runs; Fig 33 (b) shows a medium ensemble data with 50 simulation runs; Figure 33 (c) is a large ensemble data with 156 simulation runs. Each run in these examples contains 20 sample points.



(a) 20

(b) 50

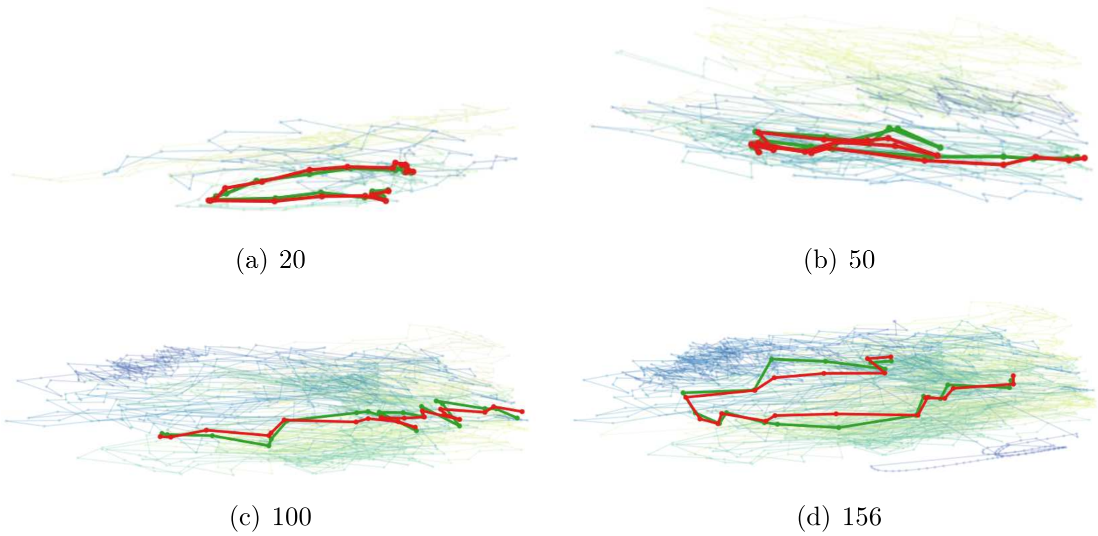
(c) 100

(d) 156

Figure 34: Sample result of inserting a new member into the ensemble visualization using the average locations of closest 4 members. The opaque time line in both figures is the new member. The green time line is our result and red is the pre-calculated position using MDS.

As we can see from the result, the closest histogram location may not be the best choice. We then use the average of n-closest histogram locations as the position for the new member. Figure 34 shows the results of the described approach. We choose 4 closest locations based on our experiments. But in some cases, the closest n locations may still be very different from the target histogram, we further set up threshold to control the number of n to achieve better results. Any histograms whose difference is larger than this threshold will not be taken into consideration of the final location calculation. Figure 35 displays the results of using threshold to improve the n-closest location average method.

From these results, the red and green lines are very close to each other which means our method is effective no matter the size of the data.Table 2 shows the time used by our algorithm and MDS. We test our approach on 4 different sizes of the storm surge



(a) 20

(b) 50

(c) 100

(d) 156

Figure 35: Sample result of inserting a new member into the ensemble visualization using thresholds. The opaque time line in both figures is the new member. The green time line is our result and red is the pre-calculated position using MDS.

dataset from 20 days to more than 150 days. Each day is sampled into 20 sample points which means each individual time line consists of 20 points. The improvement of time used is linear to the number of runs considered. There is a huge advantage when the dataset is large, while small dataset it takes about the same amount of time.

Table 2: Time comparison of our methods and MDS.

Dataset	Total Time Step	Our method (seconds)	MDS (seconds)
Storm	20	4	5
Storm	50	8	23
Storm	100	18	48
Storm	156	50	83

There are some failed cases of this approach when the new histogram is quite different from the other simulation runs or it is located near the border, average or closest locations will not work properly. As we can see in Figure 36 left, when there is only several simulation runs, the result positions are pulled towards the major part

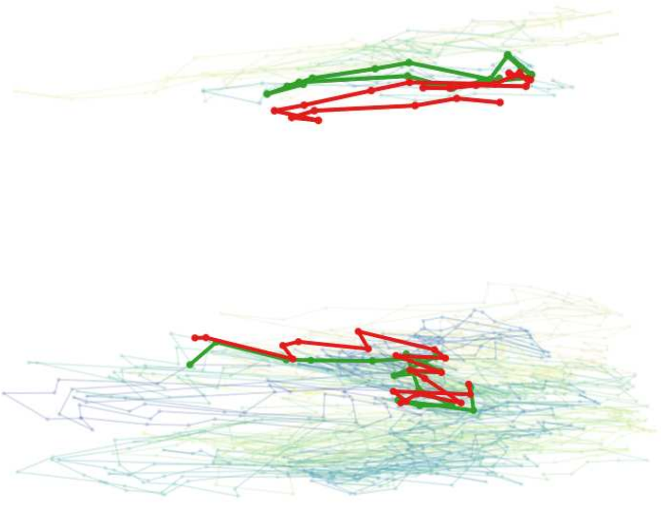


Figure 36: Failed cases of ensemble Acceleration.

of where the time lines are located. The right side of Figure 36 shows insertion into a large number of runs, but the inserted run is on the border of the entire time line, the result is shifted too.

5.5 Conclusion

In this chapter, we present our ensemble time line visualization approach. Our method first generates a single time line for the ensemble dataset. From this time line, scientist can visualize individual or overall time lines. All the members in the ensemble dataset is visualized at the same time. Easy comparison can be made by looking at multiple time lines. Further inspection and analysis can be performed if outlier or abnormalities are observed. This is an efficient way to find features within an individual member and between different ensemble members.

The local histograms of virtual words contain rich and meaningful information.

We use the properties of the virtual words distribution and can easily find method to improve our visualization results and avoid unnecessary calculations and processes. The analysis of virtual words helps us in generating more focused representation and accelerating the visualization process.

Visual cluttering is unavoidable when the size of the ensemble member is large. We solve the visual clutter problem by comparing the histograms between members and automatically generate groups for members, highlighting the representative member and hiding the others. We also can generate different views and interactions for scientist to explore, study and analyze within one member or between members using basic information visualization techniques.

Since simulation data is produced everyday, we propose an approach to accelerate the process of adding the new member into the existing time line visualization. One advantage of this is when the new data is produced, we do not need to calculate it with all the other members through MDS again. We utilize the previously generated time line and place the new incoming data based on its distance to the previous members in the ensembles. No complicated calculation are need in this case and an abstract representation of the new member are generated for scientists to do direct comparison which is essential when dealing with large amount of simulation runs. Our approach provides a suitable calculation efficiency to solve the issue of the newly coming simulation member.

CHAPTER 6: ANIMATED VISUALIZATION

6.1 Introduction

In this chapter, we present an approach to generate animated visualization as digital stories, which can effectively demonstrate temporal events by mimicking the composition and transition of storytelling techniques.

Animation has been used extensively in visualization. Studies from both research and education have reported that users often describe animation as fun and exciting, which elicits the usage of animation for effective visualization. Especially for time-varying data research, where temporal evolutions are often focused, animation becomes a natural way to represent and analyze the characteristics of data changes across time. For example, storm surge research studies the behavior of hurricanes and their impact on water volumes, which are two important events over the entire duration of the storm. It is obviously not efficient for users to analyze data features and relationships with static visualization for each time step. Therefore, exploring suitable ways to generate animation and incorporating animation in the interactive exploration process are valuable for time-varying data visualization.

However, existing visualization systems only provide limited support to generate animations. They are often produced with limited capabilities through adjusting rendering configuration, such as allowing users to record snapshots, adjust rendering parameters, and blend time segments. It is time-consuming for users to determine and modify all these parameters before reaching a satisfying animation result. Also, since these rendering configurations do not imply event semantics directly, current approaches can be very difficult for non-professional users. Therefore, useful storytelling techniques should be considered to describe event features in the visualization, so that the produced animations are easy to understand and follow. Up to now, there has not been such an approach to connect the generation of animation with the description of temporal events.

Our approach considers general tips of both storytelling and story writing. For example, as shown in Figure 37, a good story may include a plot with several stages, such as exposition, conflict, rising action, climax, a falling action and final resolution [103]. Stories often describe a subject from different aspects and switch topics smoothly. Similarly, our approach describes a temporal event from different aspects and scales through the concept of an event graph (Figure 38), which abstracts a time-varying dataset as a graph of related event features. Animations are then generated by searching for similar narrative structures shown in Figure 37. Our approach also allows interactive generation and modification of digital stories based on the semantics of event features.

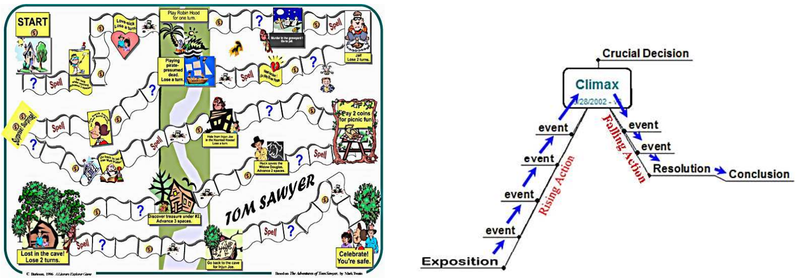

START
Love-sick Lose a turn.
Play Robin Hood for one turn.
Murder in the graveyard! Go to jail.
Spell
Jail Lose 2 turns.
Playing pirate-presumed dead. Lose a turn.
Surprise! Surprise!
Pay 2 coins for picnic fun!
Hide from Injun Joe in the Haunted House! Lose a turn.
Go back to jail & free Muff Potter!
Huck saves the Widow Douglas. Advance 2 spaces.
TOM SAWYER
Discover treasure under #2. Advance 3 spaces.
Lost in the cave! Lose 2 turns.
Go back to the cave for Injun Joe.
Celebrate! You're safe.
© Burleson, 1996 A Literary Explorer Game
Based on The Adventures of Tom Sawyer, by Mark Twain
Crucial Decision
Climax
event
event
event
event
Rising Action
Falling Action
event
event
Resolution
Conclusion
Exposition

Figure 37: Example representations of narrative structures.

Specifically, animations are designed for both summary visualization and interactive exploration of time-varying datasets. Given an event description, we first automatically construct an event graph that abstracts event features from the entire time duration as nodes and their relationships as links. The event graph is built by exploring suitable segmentation of event durations according to different data features, thereby embedding a tree-like hierarchical structure. Narrative structures are built by exploring suitable starting points and search strategies in the event graph. We provide automatic graph search algorithms as well as interactive modification methods. According to different stages of a narrative structure, we automatically determine all the rendering parameters to highlight event features and construct story transitions. We also integrate this animation generation approach into the interactive exploration process of time-varying datasets, so that suitable animations can be synthesized promptly to provide more comprehensive event visualization. We demonstrate with a storm surge application that our approach allows semantic visualization of time-varying data by generating suitable animations to describe various event features.

The main contribution of our automatic animation generation approach is that it employs animation as a summary and exploration tool to enhance time-varying data visualization. The key concept of our approach is the event graph, which characterizes temporal patterns from different aspects and scales in a time-varying sequence. Closely related to the event graph, several options of constructing narrative structures are explored, ranging from automatic graph search strategies to interactive modification methods. Our approach allows semantic generation of animations, as users can

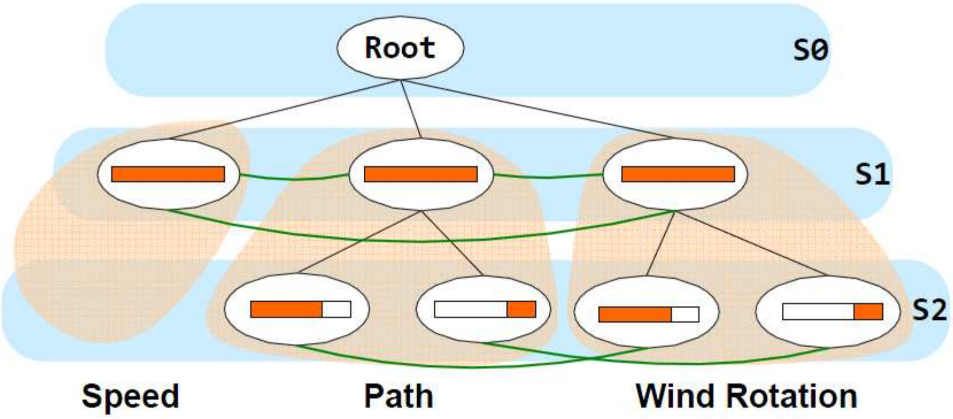


Figure 38: An event graph for the behavior of hurricane eye. The event is described from three feature aspects at different scales. The orange bar inside each node represents its time duration. The scale 0 contains a virtual root that connects all the feature aspects. The black edges are tree links which indicate child/parent relationships of features belonging to the same aspect. The green edges are relation links which describe similarities of event features from different aspects.

easily produce or modify animations by selecting feature aspects, event details, and options of digital storytelling. We also provide a composition solution for animations by determining optimal parameters automatically with a narrative structure. Partial of this chapter is published in [159].

6.2 Approach

In this section, we first present the concept of event graph. Then, we describe our approach to construct an event graph and build narrative structures given a user-specified event. We further present our automatic methods to determine all the required rendering parameters and generate animations with smooth transitions. In addition, we introduce two usages of animations for time-varying data visualization.

6.2.1 Event Graph

The purpose of the event graph is to abstract various features of an event in a time-varying dataset. We choose the form of a graph, which represents event features as nodes and node relationships as links, since it is close to the narrative structures used in storytelling techniques, as shown in Figure 37. With such an event graph, we can generate animations with narrative structures built by exploring suitable starting points and graph search strategies. Since most of our narrative structures are composed of linked node pairs, which correspond to semantic transitions between related event features, we can convey different feature aspects or scales of an event smoothly in the resulting animations.

For example, Figure 38 shows an event graph of a hurricane eye in a time-varying storm surge dataset. The behavior of the hurricane eye is described from the aspects of moving path, moving speed, and wind rotations around the eye region. A reasonable narrative structure is to describe these three feature aspects respectively, like introducing an event from different viewpoints; and each aspect from scales of low to high, like starting with an overview and gradually getting into the details.

We can represent an event graph G as follows:

$$G = \{\{nodes\};\ \{tree\ links\};\ \{relation\ links\}\} \tag{12}$$

where an event graph consists of three sets of information: nodes for representing event features from several aspects and at different scales, tree links for indicating the child and parent relationships of nodes belonging to the same feature aspect;

relation links for indicating the similarities of time durations of nodes from different feature aspects.

Specifically, the nodes without any children correspond to basic events, such as a straight moving path or a constant moving speed. The nodes with children correspond to complex events which are composed of multiple basic events. Every event graph also contains one virtual node that connects all the features aspects. The number of nodes in an event graph is related to both the event complexity and detail levels according to user interest. Each node has several components: feature aspect, time range, a score measure to indicate the significance of this event feature, a list of tree links, and a list of relation links.

There are two types of links: tree links and relation links. The tree links represent the relationships of complex events and basic events belong to the same feature aspect. The duration of a parent node always contains the durations of its children nodes. The relation links describe the temporal similarities of nodes that are on the same scale but from different feature aspects. Each relation link has two components: a related node and a similarity value, which uses 0 to represent "the same", 1 to indicate that the time range of the current node includes the related node, -1 vice versa, and the ratio of overlapped time duration to the time range of the current node for other cases. To avoid confusion, we use relation links with value 0 most of the time.

An important feature of the event graph is that each event graph embeds a tree-like structure that organizes all the relevant event features. As shown in the graph example in Figure 38, each feature aspect is represented as a tree branch. The scale 0 represents a virtual node as the tree root. The scale 1 always represents event features from the entire duration of a time-varying dataset. Starting from the scale 2, the nodes are calculated by segmenting the time range of parent nodes according to the characteristics of data changes. With this tree structure, we can locate different levels of event details according to the node scales.

6.2.2 Constructing Event Graphs

We construct an event graph by following the embedded hierarchical tree structure. Initially, given an event and its feature descriptions, we can first build a basic tree structure with the virtual root at scale 0 and a node for representing each feature aspect from the entire time duration at scale 1.

Starting from scale 2, we segment the duration of each node on a parent scale until reaching basic events. As shown in Figure 39, the moving speed of the hurricane eye is reducing constantly, therefore it is a basic event and only contains one node. The river surface elevation and the wind rotation of the hurricane eye contain more variation. They are represented by a sub-tree in the event graph. Our approach to time segmentation is through measuring the dissimilarities of time steps automatically. For every feature aspect, we compare each pair of adjacent time steps and quantify their differences as one value, which can be shown as the curves in Figure 39. We detect if this node describes a basic event by testing if all the dissimilarity values are below a user-specified threshold. The segmentation process stops when reaching a basic event; otherwise it continues to higher scales.

The segmentation is achieved by searching for the cuts that divide a time duration into several segments, where the dissimilarity values are more evenly distributed.

Specifically, we calculate the gradients of dissimilarity values, smooth them with a Gaussian operator, and apply Sobel edge detection [131]. The locations with the maximum results are the cuts to segment a time duration. For storytelling effects, we limit the number of children for each node to 3, so that we can keep an evenly distributed event graph.

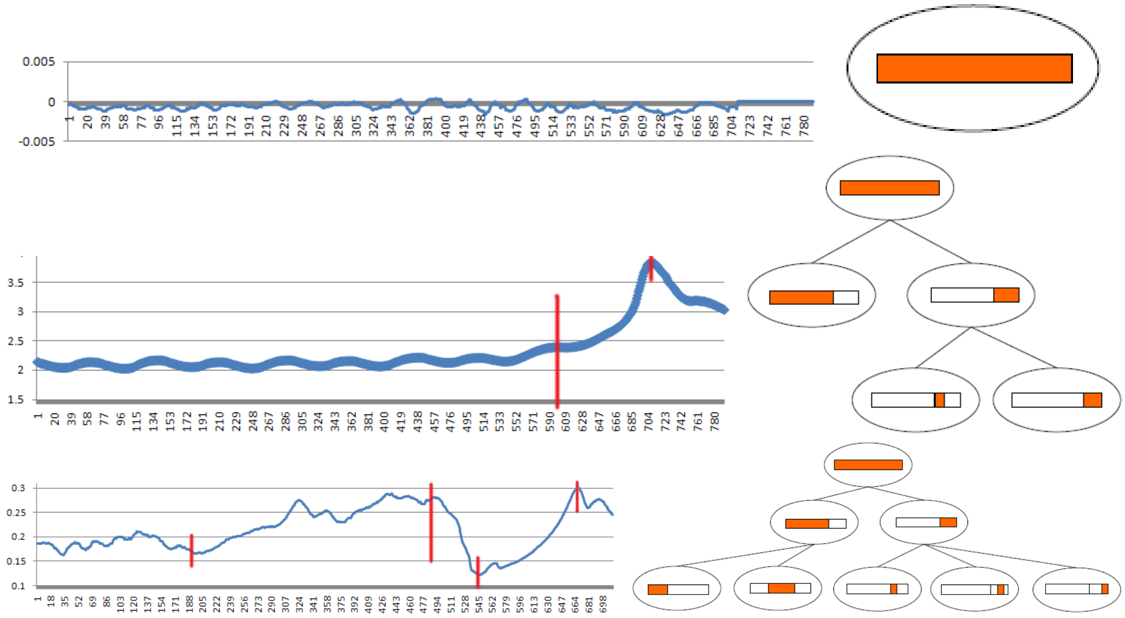


Figure 39: (left) The temporal dissimilarity curves for moving speed of the hurricane, river surface elevations, and wind rotation of the hurricane eye. Our segmentation results are shown as the red lines. Longer lines indicate segmentation on lower scales. (right) Constructed sub-trees.

For example, for the event of the hurricane eye, we first locate the path of the eye as the weighted center of wind magnitudes for each time step.

The moving speed is measured as the shift of eye location in adjacent time steps. For the wind rotation, we first identify a region around the hurricane eye with high wind magnitudes, which is shown as the blue region. Then we sum the wind angle at each vertex of all the triangles within this region. Since the hurricane speed is roughly constant during the entire duration, this branch only has a leaf on scale 1, while the moving path and wind rotations are further divided to higher scales.

To match event features from different aspects, we combine them during the construction process of the event graph. Specifically, we determine if two nodes with the same duration, but from different feature aspects, can be segmented jointly. We achieve this by measuring the distributions of their dissimilarity values. The dissimilarity values from the involved time durations are concatenated and treated as one high-dimensional vector. We combine two feature aspects if the dot product of their normalized dissimilarity vectors is smaller than 10% of the time duration. We further merge their dissimilarity values for the segmentation process. Otherwise, the conclusion is that these two aspects are very different and should be segmented separately. For example, the moving path and wind rotation of the hurricane eye are segmented jointly on scale 2.

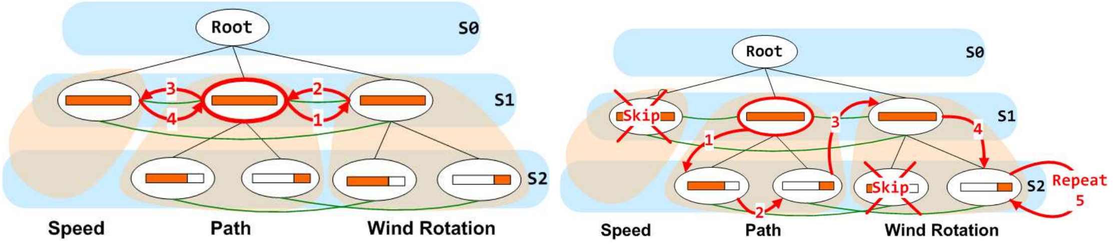
The relation links are constructed for each scale to describe the relationships of node durations from different feature aspects. For scale 1, since the durations of all the nodes are the same, there is a relation link with value 0 between every node pair. Starting from scale 2, we measure the relationships of node durations and assign corresponding values. For example, there are two relation links between the two nodes under moving path and wind rotation aspects in Figure 38.

We also measure a significance score of each node based on the multiplication of its temporal duration, region size and dissimilarity variance of an event within the node duration. The region size is measured as the accumulated area that covers the event for the entire duration. For example, in Figure 38, the moving path covers all the regions where the hurricane passes, while the wind rotation is only related to the small region of hurricane eye. The dissimilarity variance is calculated through the dissimilarity curve, shown in Figure 39. For each scale, we further normalize all the scores by dividing the largest value, so that they can suggest the significance portion on this scale. The virtual root is always assigned a score value 0, meaning that this node should be skipped. We also balance different feature aspects by multiplying a user-assigned scale with the dissimilarity variance. For example, we magnify the scores of wind rotation by 10 to make them comparable to the scores of other feature aspects. The significance results describe the events in the example of Figure 38 well. On scale 1, the moving path has the highest score value, 1.0, since it describes an event that involves the entire data region. The score value of moving speed is the lowest, 0.1, since it has to do with details which are not obvious when we consider the entire time duration.

Scientists often have some domain knowledge of the data under study. For example, in the storm surge study, scientists are interested in the behavior of hurricane and effects of the hurricane on ocean and river volumes. Therefore, we can build a set of event graphs before visualization. Figure 41 shows another example result of an event graph built from a storm surge dataset: bumps and splats in the river.

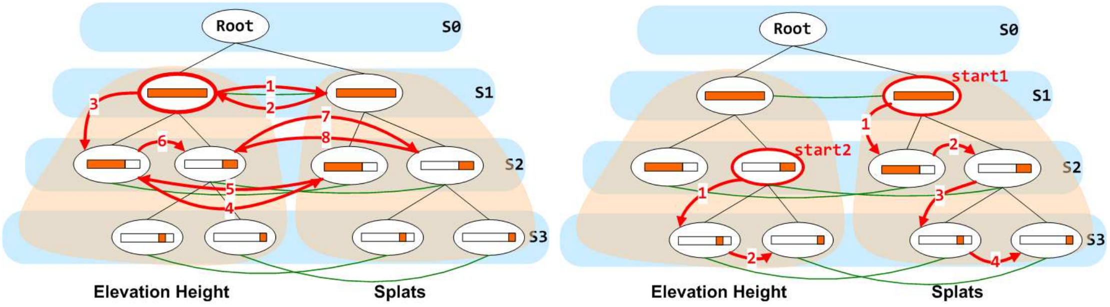
6.2.3 Building Narrative Structure

A narrative structure is a plot that determines the sequence of event features in an animation. It is built by searching for a suitable node as the beginning and a search strategy that traverses the event graph and satisfies user-assigned storytelling



Relation search Interactive depth-first search

Figure 40: The event graph of the hurricane eye and 2 narrative structures, shown as the red links. The node with red boundary is the starting point. The top shows the result of a relation search algorithm on scale 1. The bottom shows the result of a depth-first search algorithm below scale 3. Interactions are also applied to skip the feature aspect of speed and one node of wind rotation, as well as repeating the end segment of wind rotation aspect.



Relation search Interactive visualization

Figure 41: The event graph of bumps and splats and 2 narrative structures, shown as the red links. The node with red boundary is the starting point. The top shows the result of relation search below scale 3. The bottom shows the result of interactive visualization, where users first choose to visualize the splat feature globally and then select the feature of elevation height close to the end of the time range.

options. Several automatic schemes as well as interactive modification methods are explored to generate narrative structures.

The starting node is preferably the most significant node on the lowest available scale, which can serve as an overview to introduce the major event feature. For example, for the event of the hurricane eye in Figure 40, we compare the scores of the three nodes on scale 1, and select the moving path node as the starting point. This is the same beginning for the two narrative structures built from different strategies in Figure 40.

The path of a narrative structure traverses the rest of the relevant nodes in the event graph. A path is composed of tree links and relation links in a specific order. Traversing along the tree links is identical to describing additional event details. Traversing along the relation links equals the description of related event features from other aspects.

These two links help us to achieve smooth semantic transitions in the storytelling.

To build suitable narrative structures, we design three criteria for the graph search strategies.

1. An event is composed of an overview at the beginning and smooth transitions to event details.
2. Related features are introduced sequentially according to their score values or temporal order.
3. No node is repeated unless specified by the user.

We can utilize some existing algorithms of graph search, such as breadth-first search and depth-first search [36], to generate our narrative structures. These two search algorithms meet our design criterion 3 directly. If we always start from a node on the lowest available scale, we can also meet our criterion 1. According to criterion 2, the only addition we make is to determine the search sequence of children nodes given a parent node. We can choose to follow the order of their temporal durations or the order of their score values.

Since we introduce the relation links to the event graph, we also develop a search strategy called relation search. Specifically, given a starting node, we follow the depth-first search algorithm. The main difference is that if the current node has relation links, we first traverse the nodes connected by relation links instead of exploring tree links. This means that we do not perform depth-first search on relation links. Examples with the hurricane and splat events are provided in Figures 40 and 41.

The design of our event graph allows easy user interaction and modification on narrative structures. For example, we provide the following interaction methods.

1. Selecting feature aspects

   Users can select different event aspects to visualize and the branches that are not selected are ignored in the searching process.

2. Selecting of level-of-details

   Users can adjust the levels-of-detail of animations and only the nodes on involved scales are traversed.

3. Skip

   Users can choose to ignore some feature aspects at certain levels (e.g., if they are already familiar with this data portion). These nodes are not considered during the graph traversal.

4. Repeat

   A portion of a narrative structure can be repeated for emphasis. We add the repeated portion to the narrative structure.

5. Selecting starting point

   Users can select the starting node. The branch containing the starting node will be treated as mostly important.

6.2.4 Automating Animation as Storytelling

To avoid unnecessary user interaction, our approach determines all the rendering parameters automatically. The parameters are calculated for each node in the narrative structure. We also provide a transition between adjacent nodes to indicate feature changes.

Our rendering parameters include: time steps, viewpoints, and data resolution. The other rendering settings are fixed in an animation to avoid confusion. For example, the region-of-interest is always highlighted in green, as shown in Figure 43. All the other regions are visualized in the same color design: terrain always in grey and surfaces always colored according to their elevation levels with a blue-to-red colormap. All these detailed rendering parameters are automatically determined based on our experiments. For professional users, an interface is also provided for adjustments.

The number of time steps selected for each node is proportional to its score value. The data loading and rendering times for each time step are approximately the same, assuming we use the same data resolution for all the time steps belonging to this node. Therefore, the number of time steps is linear to the allocation of time duration.

To emphasize important features, a small number of time steps is used for basic events, while a rather large number is used for major events like the hurricane moving path. We set the maximum allowed number of time steps N_{max}, which can also be adjusted. The number of time steps for a node n with score $score(n)$ is calculated as $N_{ideal}(n) = score(n)^p \times N_{max}$, where p is a parameter to adjust the effect of score values. We treat an event evolving gradually in each node duration, so we sample time steps evenly. Assume the number of time steps contained in a node is $N(n)$. If $N(n) >= N_{ideal}(n)$, we select a time step for every $N_{ideal}(n)/N(n)$ time steps. Otherwise, we set a sleep time proportional to $N_{ideal}(n)/N(n) \times rendering\ speed$ in the program.

Several options for viewpoints are explored for our animation: sky view, local view, dynamic view, optimal view, and user-selected views. Generally, we prefer to put the event close to the center of the screen and avoid having the interesting region occluded by surroundings.

1. Sky view

   A sky view is often used in TV programs. It points to the event center, is vertical to the main event moving direction, and has a 60 degree elevation. The view zooms into a region whose size is 1.6 times the bounding box of the interested region in this time duration.

2. Local view

   A local view produces the effect that viewers are standing close to the ground, which can be found by reducing the elevation degree of the sky view to the range of [0, 30].

3. Dynamic view

   The dynamic view follows the path of the event and points to the object center at each time step. We smooth the path to avoid the shaking camera effect. Similar to the sky view, the view zooms into a region whose size is 1.6 times of the bounding box of the interesting region in this time step.

4. Optimal view

   The optimal view can be selected according to the object shapes and overlap relations, such as through maximizing data variations [93]. It is the same for each event node. We simplify this step by running 2D principal component analysis (PCA) [149] on the object center locations from the entire node duration. The optimal view is calculated as rotating the second axis of PCA to an elevation of 60 degree. The view zooms in the same way as the sky view.

5. Selected view

   Our system allows users to choose their desired viewpoint by rotating and zooming the scene.

Data resolution is selected according to the performance requirement. Low resolution is used for interactive exploration. In our experiment, our machine can support a reasonable performance (around 0.5 second per time step) when using data contains 520114 polygons.

To provide smooth transition between adjacent nodes, we introduce three additional renderings of the event location, duration and graph. They are rendered automatically to provide useful information about an event feature. As shown in Figure 43, we use the top right corner to show the location of an event, which can be calculated with

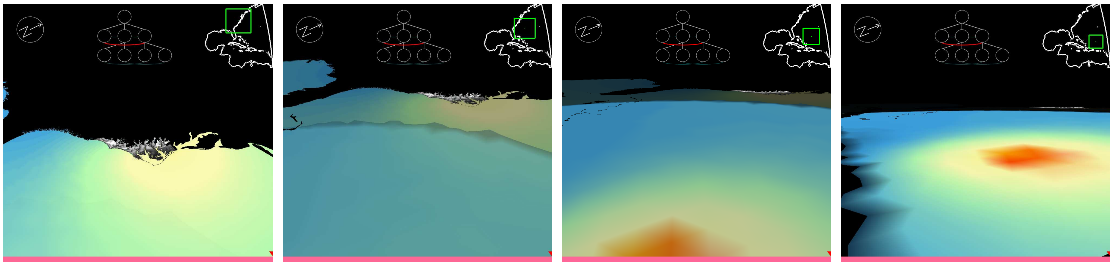


Figure 42: A transition example. Four snpshots are provided from the transition of wind rotation to the moving speed on scale 1, which is indicated by the red curve on the event graph on the top. During the transition, the viewpoint is smoothly rotated and the rendered data are automatically blended to produce a rotating and fading effect.

the camera setting. The time duration is shown on the bottom of the screen, which renders the duration of the current event node in pink. The current time step is also specified with a red arrow above the time window automatically according to data names. A simple event graph is rendered on the top with current node or link in red. These two renderings change automatically according to the node information. When the node switches, we change the scene by linearly interpolating the ending viewpoint of the previous node and the starting viewpoint of the next node, and then start the animation for the next node. We also blend the data from the two transiting nodes in the rendering to produce a simultaneous rotating and fading effect. Figure 42 provides such an example.

Animated Visualization

We provide two options of digital storytelling using animations: summary and interaction modes. The summary mode automatically produces an animation to summarize the relevant event features in a time-varying dataset, while the interaction mode is designed to generate and show animations in real-time to provide more comprehensive event information than static visualization for interactive exploration.

The summary mode does not have a performance constraint, and thus we can include all the nodes of an event graph in the animation results. Any of the three searching strategies can be used to build narrative structures. We can also render datasets in their original resolution and utilize any provided methods to determine our rendering parameters. Figures 40 and 41 (top) provide examples of summary animations using relation search with different levels-of-details.

The interactive mode allows users to apply animations to enrich static visualization. Instead of requiring users to select their interesting data region from every time step, we render a time-varying dataset according to a narrative structure automatically. This setting allows users to replay an animation, stop an animation anytime, and most importantly generate a specific animation that describes their interested event features. Such interactive exploration provides more relevant information to users efficiently. Since the construction of the event graph takes time, especially for large time-varying datasets, we require users to specify the event under exploration before interaction. The event graph of any event only needs to be built once, therefore we can quickly collect all the event graphs of major events. In the storm surge application, the behavior of the hurricane and its impact on the water are two important events. We also perform data reduction before the exploration process by equally downsampling the grid meshes for all the wind field, terrain and water surfaces.

The challenges of real-time animation generation are to find a suitable starting node based on user selection and satisfy performance requirement. During interaction, users can specify their interesting event aspect and adjust the time step and viewpoint to visualize a specific data portion. With this information, we locate the matching node on the highest scale from the corresponding branch in the event graph. The time range of this node should include the selected time step and the node region covers the visualized data portion. Among our search strategies to generate narrative structures, we often choose the depth-first search and relation search algorithms, as they provide most relevant information for user interaction. Since a narrative structure is built very quickly, we can start the resulting animation without any delay. To satisfy the interactive performance requirement, we simplify choices of rendering parameters. For example, we use sky view, local view, and dynamic view instead of optimal view. We also use data in lower resolutions. Figures 40 and 41 (bottom) provide examples of animations under the interactive exploration mode.

6.3 Results and Discussions

The time-varying datasets used in this dissertation are generated by a storm surge or inundation model, ADCIRC (http://www.adcirc.org/), which has been approved for storm surge studies by Federal Emergency Management Agency (FEMA) and applied extensively for modeling storm surge in the Southern Louisiana and New Orleans areas. Specifically, we use the data of hurricane Isabel to the Outer Banks of North Carolina, USA, in 2003 [145]. Each dataset contains 792 time steps, 520,000 triangles, and 260,000 vertices. There are three attributes, terrain elevation, water elevation, and wind vector on every vertex.

6.3.1 Example Results

The attached multimedia files provide our animation results, which correspond to the narrative structures in Figures 40 and 41. These four examples demonstrate different automatic narrative structures and interactive modification results. The event graphs of two events, hurricane eye and splats, are constructed automatically based on event descriptions. In all the renderings, the terrain surfaces are rendered in gray. The water and hurricane surfaces are illustrated with a blue to red colormap, with red corresponding to the highest elevation value. The options of viewpoints are randomized in the generated animations to avoid uniform rendering effects. Example snapshots are provided in Figure 43.

The narrative structure in Figure 40 (top) is generated with the relation search strategy with the lowest details. The resulting animation provides an overview of the three event features. It first shows the moving path of the hurricane eye, which is close to a straight line, since it is the most significant feature. Second, the animation visualizes the wind rotation around the hurricane eye, which mostly increases time range. At the end, it introduces the moving speed, which is constantly reduced.

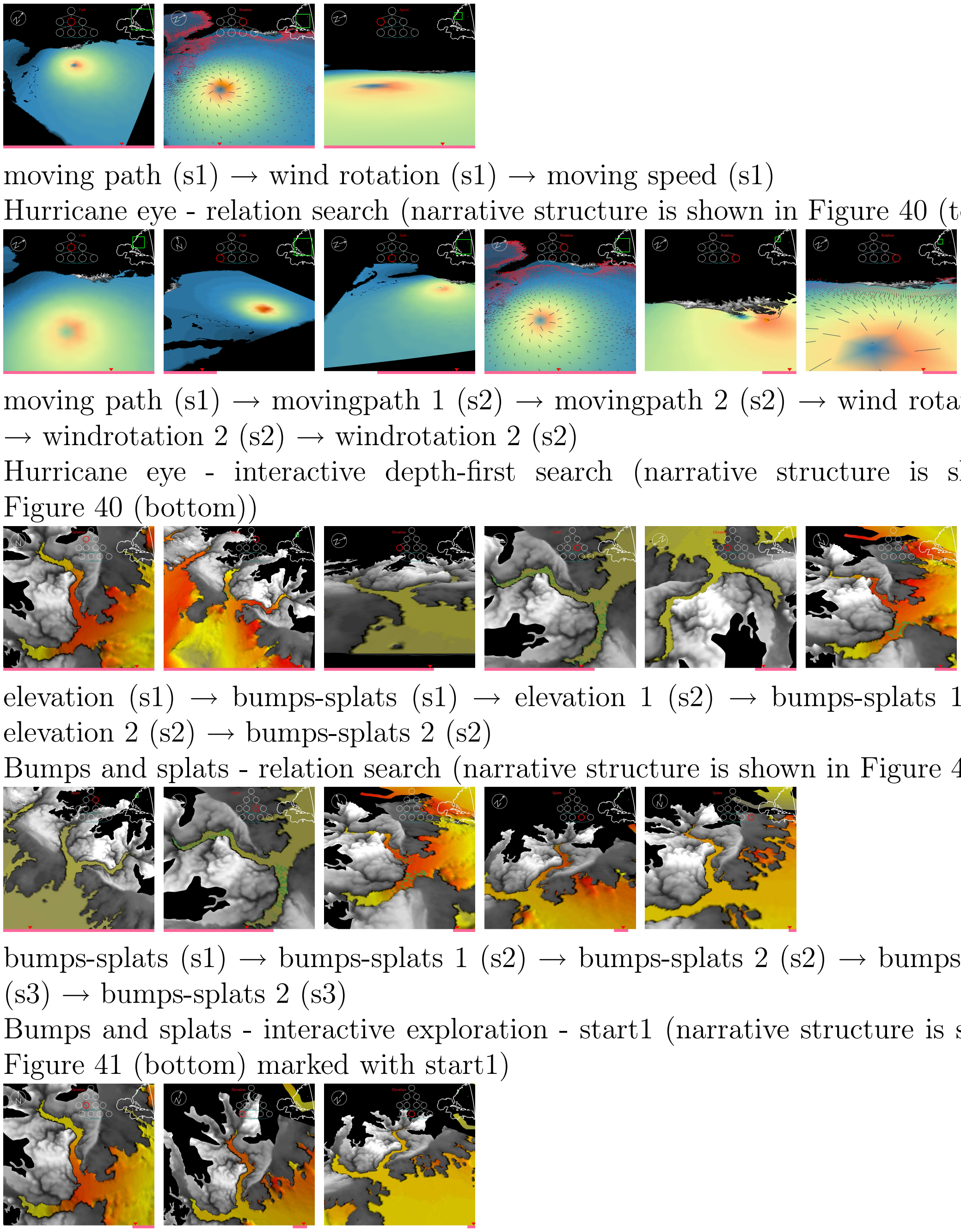
The narrative structure in Figure 40 (bottom) is generated with the depth-first search strategy. The resulting animation provides more details of the event than the first animation. It has the same starting point, which overviews the moving path of the hurricane eye. The animation then visualizes each segment of the moving path. Then, the animation overviews the wind rotation and visualizes the second half time range for the hurricane eye. Since the second half is nearer to the coastal terrain, which is more important to scientists, users can choose to skip the first half range and repeat the second half. Different parameters are selected to provide more comprehensive information on the wind rotation.

Our second example event is the bumps and splats in the rivers. The narrative structure in Figure 41 (top) is generated with the relation search strategy with middle level of details. The animation first overviews the water surface elevation height, which is the dominant feature of this event. It continues to introduce the bumps and splats in the river, which are highlighted as green dots. Then, the animation visualizes the elevation height and splats respectively, for both time ranges, where we can see a periodic feature during the first time range and the effect of the hurricane during the second part.

The narrative structure in Figure 41 (bottom) is generated with interactive exploration. We first choose to visualize the bumps and splats, so the animation starts with an overview on scale 1 and continues to visualize the data from the first and second time ranges respectively. We then choose to view the elevation height over the second time range, and the animation locates the starting node on the scale 2 and generates an animation with the depth-first search algorithm.

6.3.2 Quantitative Results

We run our algorithms on a computer with an Intel Core2 CPU 6600 at 2.40GHz and 2GB RAM. Most of the processes in our approach, construction of the event graph, calculation of rendering parameters, and data reduction, are preprocessed before interactive visualization. The running time for constructing an event graph



moving path (s1) → wind rotation (s1) → moving speed (s1)

Hurricane eye - relation search (narrative structure is shown in Figure 40 (top))

moving path (s1) → movingpath 1 (s2) → movingpath 2 (s2) → wind rotation (s1) → windrotation 2 (s2) → windrotation 2 (s2)

Hurricane eye - interactive depth-first search (narrative structure is shown in Figure 40 (bottom))

elevation (s1) → bumps-splats (s1) → elevation 1 (s2) → bumps-splats 1 (s2) → elevation 2 (s2) → bumps-splats 2 (s2)

Bumps and splats - relation search (narrative structure is shown in Figure 41 (top))

bumps-splats (s1) → bumps-splats 1 (s2) → bumps-splats 2 (s2) → bumps-splats 1 (s3) → bumps-splats 2 (s3)

Bumps and splats - interactive exploration - start1 (narrative structure is shown in Figure 41 (bottom) marked with start1)

height 2 (s2) → height 1 (s3) → height 2 (s3)

Bumps and splats - interactive exploration - start2 (narrative structure is shown in Figure 41 (bottom) marked with start2)

Figure 43: Example renderings from the animation results. Each row corresponds to a narrative structure in Figures 40 or 41. The event location of each node is illustrated on the right top corner and the node duration is rendered as the red bar on the bottom of each snapshot. The little red triangle above the duration bar indicates the specific time step of each snapshot. The texts under each row suggest the node sequence in each animation. Please refer to the descriptions in section 4.1 (example results) for more details of these animations.

depends on the selected event features. The event graphs in Figures 40 and 41 take around 7 minutes. We also determine parameters for the rendering at this stage, which is within 5 minutes. Data reduction can be achieved within an hour and it is only required once for a dataset.

The only operation during interactive visualization is the generation of narrative structures. The performance of all our search algorithms is interactive. Therefore, once an event graph is built, our approach allows users to generate and modify animations interactively.

6.4 Conclusions

This chapter presents an automatic animation approach to visualize temporal events by mimicking the composition and transition of storytelling techniques. This is achieved with the concept of an event graph, which abstracts event features from different aspects and scales, and an automatic animation process with a series of event detection, time segmentation, graph search, and rendering composition methods. Our approach also allows interactive modifications of both narrative structures and parameter selections. The resulting animations can be used in summary visualization as well as interactive exploration, which provides more comprehensive information of a time-varying dataset in a timely fashion. We demonstrate with a storm surge application that our approach allows semantic visualization of time-varying data and easy animation generation for users without special knowledge about the underlying visualization techniques.

Our approach provides a framework to generate animations to describe various temporal events. As shown in our example results, our approach only requires users to identify some high level key aspects of the time-varying phenomena and how to characterize the associated features, such as motion path, eye, and wind speed for the hurricane eye event. This information can be obtained through discussion with domain experts. As a result, in this case we have a structure that can be applied to *all* hurricane and storm surge simulations, and which also can be used to classify and correlate these different simulated phenomena. For our storm surge application, scientists have pointed out that the bumps and splats are directly related to the storm modeling technique (and may be unphysical artifacts); therefore they are very interested in characterizing and understanding these events. Thus we have a process to add important new features in a straightforward manner. This overall process of developing a rich narrative framework by considering a small number of key high level aspects can be applied generally to a range of time-varying phenomena.

CHAPTER 7: EVALUATION OF ANIMATED VISUALIZATION

7.1 Introduction

Animation has been widely used to show trends in real-life applications. It is one of the most popular choices for many end users, as it is a natural way to represent trends in scientific data containing 2D or 3D structures. In visualization of time-varying datasets, animations are often generated by connecting snapshots from individual time steps [11, 61, 95, 150, 112, 135]. Several methods have been developed for animation in visualization, including both interactive editing systems for assisting users to generate animations and automatic approaches. Akiba et al. [11] developed a template-based animation tool for volume visualization. Gershon [61] presented methods for displaying a series of blurred images in an animation loop. Viola et al. [135] presented a method to focus viewpoints automatically on features of a volumetric dataset. Woodring and Shen [150] highlighted objects in a volume dataset by applying animations with positional motions and opacity variations.

However, there have been serious concerns on the effectiveness of animation in the field of visualization. The main focus is on if the dynamic features of animations suit for data exploration and analysis applications. Dynamic visualization has its advantage over static media, especially for teaching students about dynamic phenomena. It is not as simple as an animation showing some visible phenomenon. Dynamic representations can also visualize variables that are not visible, but are spatially distributed, such as changes in atmospheric pressure or temperature on a weather map. It can also display statistical concepts such as changes in variables or computer algorithms. Dynamic visualization can also be a metaphor for abstract information and distort reality such as speeding up some process, slow down other, changing viewpoints, augmenting the visualization with cues to draw viewers' attention. But the fact is that dynamic displays are not always easy to understand. Dynamic visualization also includes non-interactive dynamic displays, interactive dynamic displays [71, 8]. Example questions include: If animations are statistically better than static or interactive visualization approaches? Does people perceive knowledge more quickly and accurately from dynamic animation than static images? How would these different visualization achieve when user performing exploration and analysis under different tasks?

Several researches, mainly from the field of information visualization, have been performed to evaluate the effectiveness of animation in visualization [70, 111]. Heer and Robertson designed animations for creating animated transitions for data graphs. Lundström et al [96] presented an animation method to convey uncertainty in medical visualization. Researcher also uses other visualization methods to represent the statistical data such as scatterplots and parallel coordinates [20, 136, 50]. Also, a large portion of work are done by comparing dynamic graphs with static graphs [52]. The evaluation results of animation are mixed, Tversky and Morrison [133] found out that animation may be ineffective when displaying events and were often too complex and too fast to be accurately perceived. Archambault et al. performed user study on mental map with animation and small multiples, and claimed small multiples give significantly faster performance but with more errors too. [13] While some researchers have found animation to be effective and helpful, others have concluded that animation is not as good as other visualizations in terms of analysis tasks although animation is fun and exciting to users [111]. Blok [21] find that users are able to extract relevant information in a monitoring context.

In this chapter, we focus on the effectiveness of animated visualization for time-varying scientific visualization and study if animations can help users to explore, analyze and understand various natural phenomena. We use storm surge visualization as a case study to present our approach extended from[159] for generating animations for various visualization tasks. Further, we have performed a formal user study to compare the effectiveness of animated visualization and traditional interactive visualization for time-varying data visualization. In this study, we attempt to evaluate the benefits of animation in presenting events and comprehending underlying data features, for which we believe animation is beneficial over interactive systems. Three typical categories of visualization tasks, presentation, exploration and reasoning, are selected for the study. Our results demonstrate advantages of animated visualization over interactive visualization.

Section 7.2 presents the detailed design, results and discussion of our evaluation. Finally we conclude this chapter in Section 7.3.

7.2 Evaluation

This study compares two types of visualization systems for time-varying data visualization: animated visualization versus a typical interactive visualization. The following presents the design of two systems and describes the details of our user study.

7.2.1 System

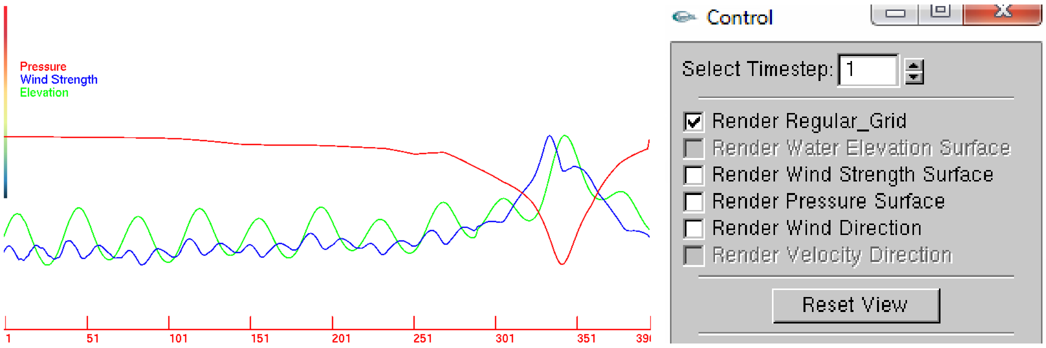
In order to evaluate how users can effectively analyze data with animations, it is important to describe the supported interaction techniques and animations and to ensure that differences in the two compared systems are understood with regards to the outcome of the study.

The interactive visualization is selected to compare with animations as it is the most commonly used approach to study time-varying scientific simulations. As shown in Figure 44, the design of the interactive visualization system also adopts typical visualization systems with three panels: a 3D rendering panel, a temporal trend panel, and a control panel.

1. Rendering Panel

   The 3D rendering panel visualizes all the involved data attributes from a selected time step. Standard interactions are provided, including rotation, zooming in/out, and viewpoint selection, selection of time step, and selection of data attribute such as the renderings in Figure 2. We only provide these standard interactions to direct subjects focus on the tasks.

2. Temporal Trend Panel



Select Timestep: 1

☑ Render Regular_Grid
□ Render Water Elevation Surface
□ Render Wind Strength Surface
□ Render Pressure Surface
□ Render Wind Direction
□ Render Velocity Direction

Reset View

Figure 44: Temporal trend window (left) and control panel (right).

The temporal trend window presents 2D curves of data fluctuations. This window is included in the system as 2D heat maps and plots of different variables are commonly used when visualizing storm surge simulations. The temporal trend curves are updated automatically according to the tasks. For example, for the task of visualizing surges, the curve of average water elevation is displayed in the temporal trend window.

3. Control Panel

   The control panel is for selecting data attributes, time steps, and reset 3D view for convenience. Our interactive visualization system only enable relevant data attributes for each task to direct the focus of subjects on the same set of data attributes and avoid the factor of user experience.

The animation system contains two panels: a 3D rendering panel and a temporal trend panel, which are the same as the interactive visualization system. The 3D rendering panel adopts the same rendering scheme for all the data attributes and time steps, in the sense that all the rendering effects are exactly the same. The main different is that for the animation system, the 3D rendering panel displays animation;

while for the interactive system, the 3D rendering panel displays the rendering from a selected time step. The control panel is disabled for the animation system to avoid the confusion of different animation effects.

7.2.2 Materials

This section describes the datasets and animations we used in the user study.

7.2.2.1 Datasets

Two storm surge simulation datasets are selected, Hurricane Isabel in 2003 and Hurricane Irene in 2011. Several nice features of these two datasets make them especially suitable for our user study.

First, the two simulation datasets are comparable from the following three aspects. Both datasets cover the same region, whose terrain model consist of more than 260,000 vertices and 520,000 triangles. A set of four key variables, elevation, wind vectors, atmospheric pressure and depth-average velocity, are provided in both datasets. The numbers of time steps from the two datasets are also comparable, Isabel contains 396 time steps and Irene contains 336. Therefore, we expect that the choice of datasets does not affect the performance of subjects in the user study.

Second, data features of the two datasets are different, which allow us to design a set of tasks for both datasets with different answers. Since subjects will not be able to answer the questions for one dataset by using the information they acquire from the other dataset, we can ensure that the order of the datasets does not affect the results of our user study. For example, the eye paths of the hurricanes in the two datasets differ completely. Irene skirts the North Carolina coast then tracks further north into New England, while Isabel makes landfall in the southern portion of the Outer Banks and track inland. Due to its path Irene demonstrates a more obvious example of back surge. Isabel has a smaller back and splats on Pungo River Canal which happens before the major surge come in. The following section describes the details of these data features.

7.2.2.2 Modeling of Data Features

To generate animations for each task, the relevant data features need to be modeled and tracked automatically. Storm surge simulations provide complex 3D scenarios which simulate the behavior of water elevation under different parameters, such as wind forces and atmospheric pressure. We concentrate on two data features that are important to scientists.

1. Back Surge Modeling

   Back surge is a very interesting event which appears frequently a lot near North Carolina coast, specifically near Outer Banks. Considering the direction of the hurricane, back surge can be found after the eye has passed and the general direction of the water velocity is directly or perpendicular to the storms direction. In other words, the back surge travels the opposite direction of normal surges. Figure 45 provides an example of back surge modeling.

   We extract land boundary vertices by comparing its height with his neighborhoods. Assume negative height value is below the sea level and positive value is above, if one vertex is below sea level and one of his neighbors is above, then we consider it as a land boundary vertex and vice versa. After we extract all the land boundary vertices, we further select boundary vertices on outer banks and calculate the directions of these vertices. For each vertex on Outer Banks, we build a small bounding box and all vertices the bounding box are connected to the targeted vertex to create multiple lines. The average direction of the direction perpendicular to these lines is considered as the out-going direction. In Figure 45, the grid of storm surge model is shown in (a), with blue color for the vertices, red color for extracted land boundaries, and green vectors for the directions of out-going waves on Outer Banks. Figure 45 (b) shows the elevation changes along the time on all the boundaries. The y-axis represents the elevation height.

The direction of the tides is simulated with the variable, depth-average velocity. We then use these two angles to determine whether the tides are traveling into land or going out of land. If the angle between tide direction and Outer Banks out-going direction is smaller than 90 degrees, we consider it as water coming out of land as back surge; otherwise, water is coming toward land. We use hue-preserved color [32] to represent the angle between current wave direction and North. Green means wave traveling towards West while orange means East in Figure 45 (b). The darkest green means the tides are traveling directly to west while orange means tides traveling to east.

We consider water coming out of land as positive value and water coming to land as negative value. Then we accumulate the +/- values into local time windows to find out a temporal trend of how the water is traveling along the time. Figure 45 (c) shows the aggregated values of local time windows. The vertices which have back surge features are clearly with very high value. Using these information we can locate the back surge.

2. Inundation

   Emergency response department and decision makes are particularly more interested in inundation timeline and level. Inundation is a simple but important feature in the simulation. It is crucial in the decision making of evacuation. We compare the elevation height of each time step with the height of land in North Carolina and record the inundation area of each time step. We also calculated the inundation possibilities for the vertices of North Carolina based on its location to the land boundary and its height.

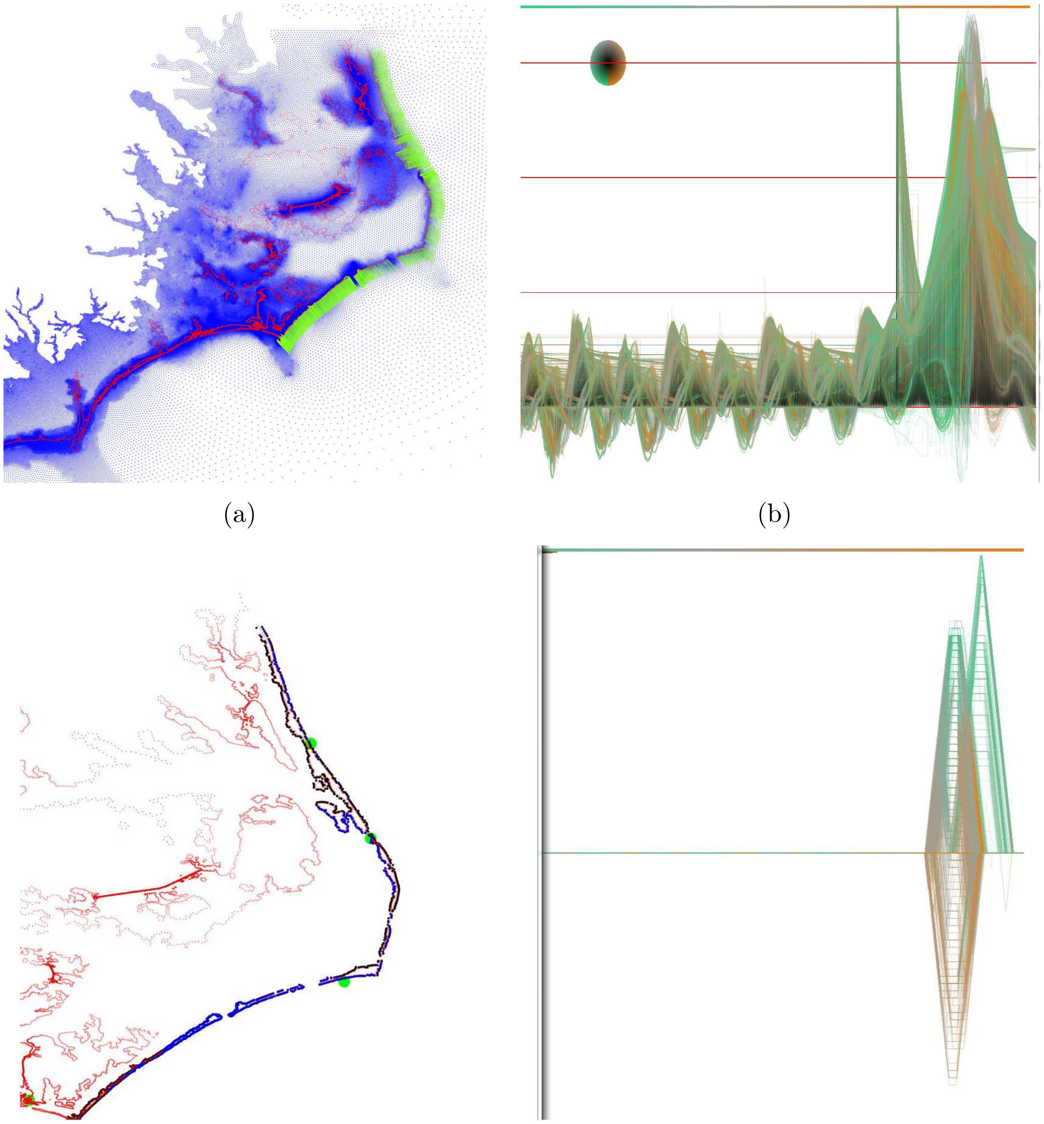
3. Hurricane path

   The calculation of hurricane path is straightforward. First, we setup a background image without wind. At each time step, we use the rendering result to subtract the background image, where we can find the location of eye with manually assigned thresholds of RGB values. After calculating the eye position, we smooth the path using a Gaussian function.

   Based on the hurricane path we extracted, we can further calculate the speed and locations of hurricane eyes along the path. Figure 46 shows a sample result of an extracted path.

7.2.2.3 Animation Preparation

We use the automatic approach in Chapter 6 to create animations for the user study. All the animations follow several design principles.



(a)

(b)

(c)

Figure 45: Model of back surge. (a) shows the vertices in the grid. (b) shows the elevation changes and tides directions. (c) shows the aggregated window values with corresponding locations rendered in blue.

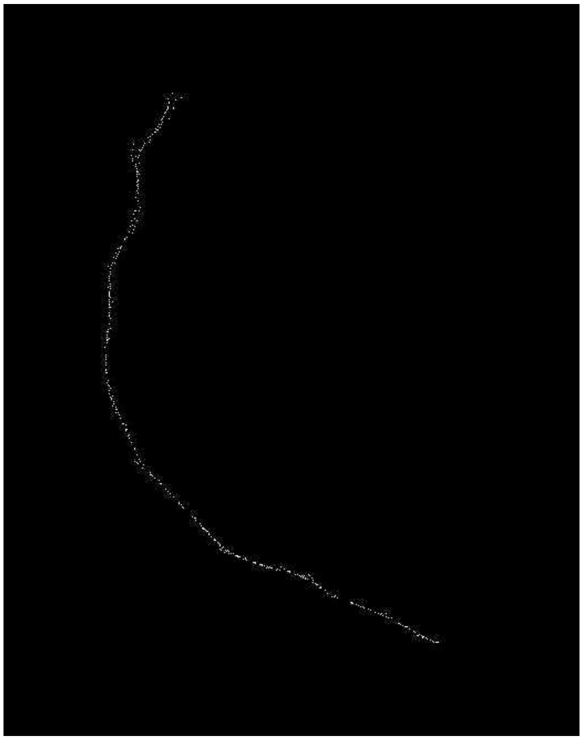


Figure 46: Result path of hurricane Irene extracted from simulation.

- Starting with an overview of entire duration
- Ending with focused view in relevant time duration
- Including all relevant data attributes in at least one segment of the animation sequence

These principles are well-accepted in visualization communities as they are consistent with the procedure of data exploration.

For each task, an animation is generated based on the assigned events and features. Details of the animations are provided for each task in Section 7.2.6.

7.2.3 Hypotheses

According to the properties of animation and interactive visualization, we postulated three hypotheses for experiments.

- H1: Animation would be faster than interactive visualization for presenting and exploring dynamic events.
- H2: Interactive visualization would be more accurate for exploring specific data features.
- H3: Animation could be more effective in identifying events than interactive visualization.

7.2.4 Experiment Procedure

Before the study, a training session is provided. Several rendered snapshots of each variable were displayed to the subjects. A demo animation is also played. Each subject is given several minutes to learn the interactions and get familiar with the system.

The order of experiments is randomized. Each task contains two sessions, with one session for animation and another session for interactive visualization. The datasets used in the two sessions are different. Both the order of animation / interactive visualization and the order of datasets are randomized in the study. The tasks are all independent and they are ordered the same throughout all the experiments to keep consistency.

The procedure is the same for each task session. Subjects are given time to read and understand the task first. For session with animation, an animation of a randomly chosen dataset will be displayed. During this animation stage, subjects watches the animation repeatedly until he/she finishes the task. They are allowed to use functions as fast forward, rewind and play back as they wish. For session with interactive visualization, subjects will use the system to finish the task. The irrelevant variable selections are disabled in order to make the study fair. Answering sheets are provided with task questions on them. Several tasks are with color figures and charts. The participants write down answers on answering sheet with task questions.

All experiments are conducted in the same room and on the same machine. The resolution of the display is 1920*1280.

7.2.5 Subjects

Participants are undergraduate and graduate students from the Computer Science department. There are 12 subjects (10 male, 2 female) with an average age of 26 (maximum 29, minimum 19). Subjects are asked to work as quickly and accurately as possible. No time limit is given. Both the results and completion time of each task are recorded.

7.2.6 Tasks

The user study consisted of three categories and two tasks in each category.

7.2.6.1 Category 1 - Representation

One important usage of visualization is to represent data as represent the currently happening events. Other user studies also include representation tasks [111]. The intention of these tasks is to examine how well subjects could understand from the animation or interactive visualization.

Task 1: Use the interactive visualization system or animation to visualize the hurricane. Draw the path of the hurricane on the map.

Related to this task, two attributes, atmospheric pressure and wind strength, were

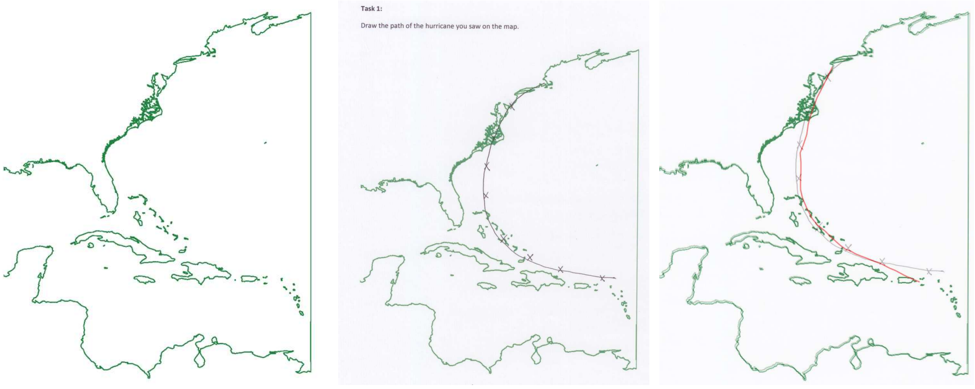


Figure 47: Left: A map was provided to subjects to draw the hurricane path; Middle: An example of a subject answer. Right: Overlapped image of ground truth image with subject answer. This image was then used to compute a pixel difference between these two.

shown in the temporal trend panel. The animation consisted of two phases; each phase displayed the hurricane wind strength and atmospheric pressure with a Bird's-eye view of the whole grid of the entire time duration respectively.

Task 2: Use the interactive visualization system or animation to visualize the inundation areas. Mark all the inundation areas inside the blue circle.

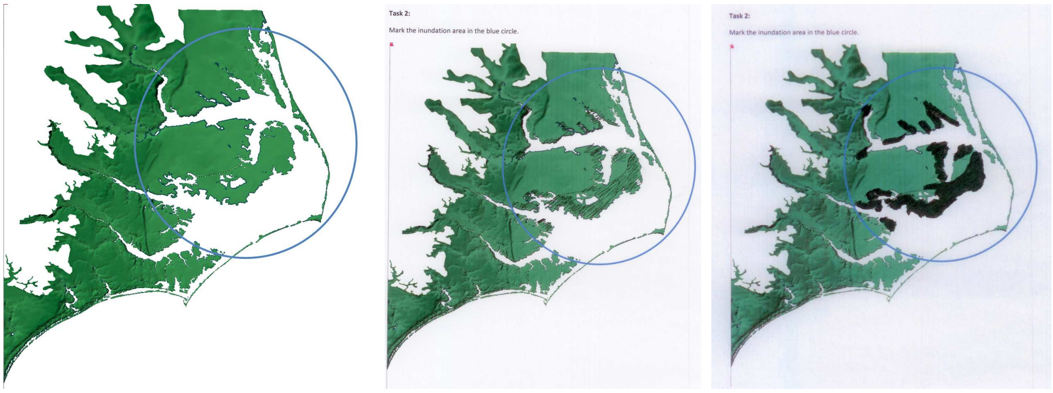

Task 2:
Mark the inundation area in the blue circle.
Task 2:
Mark the inundation area in the blue circle.

Figure 48: Left: A map was provided to subjects to mark the inundation area; Middle: An example of a subject answer. Right: Processed subject answer image, the user answers were painted black. This image was then compared with the ground truth image.

Elevation was the only attribute related to this task and was enabled in the control panel. The average of elevation of every time step was shown in the temporal trend window. Animation of inundation along North Carolina was shown to the subjects including three phases: a whole time overview of the North Carolina coast; the second phase showed a more focused view on areas around Outer Banks, the Pamlico and Albemarle Sounds. They were asked to look for storm surge time duration, where the highest possibilities of inundations were. The last phase displayed a view on North Carolina coast of the time after the surges.

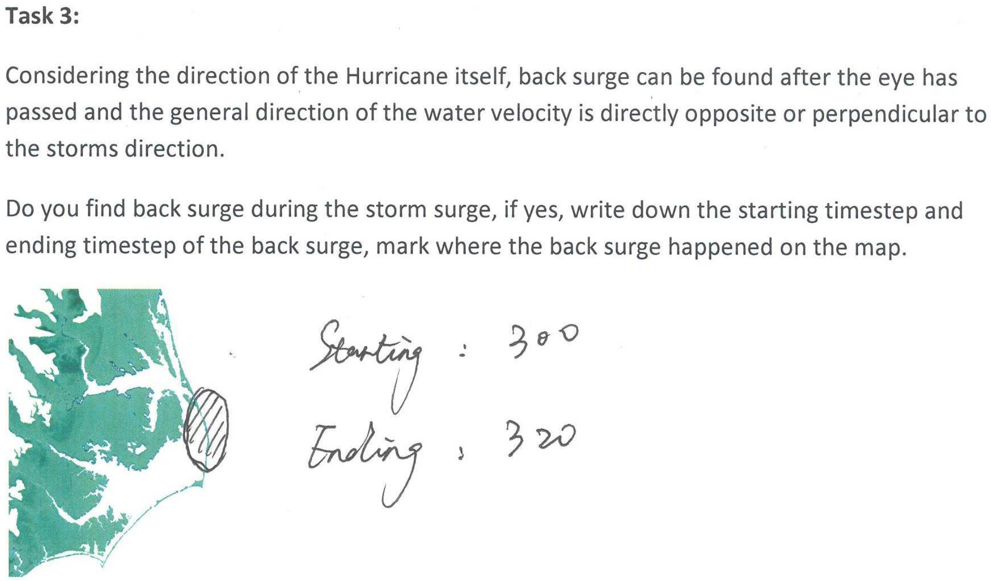
7.2.6.2 Category 2 - Exploration

This category focused on analyzing data with animation and interactive visualization. The subjects were asked to explore if certain features or events existed in the simulation. In addition, the subjects were asked to explore the relationships between different variables, such as wind vectors and elevations.

Task 3: Do you find back surge during the storm surge, if yes, write down the starting time step and ending time step of the back surge. Mark where back surge happened on the map. The map was the same as task 2 without the blue circle.

In this task, the subjects were given the definition of back surge and the average value of elevation of every time step was displayed. Subjects were asked to find specific time steps and locations of back surges beginning and fading away. Since back surge was one of the most important features found in the analysis of our data, this task would focus on searching for back surge. A sample answer from subject was shown in Figure 49.

The animation contained three phases: the first phase showed the overview on the



Task 3:

Considering the direction of the Hurricane itself, back surge can be found after the eye has passed and the general direction of the water velocity is directly opposite or perpendicular to the storms direction.

Do you find back surge during the storm surge, if yes, write down the starting timestep and ending timestep of the back surge, mark where the back surge happened on the map.

Starting : 300

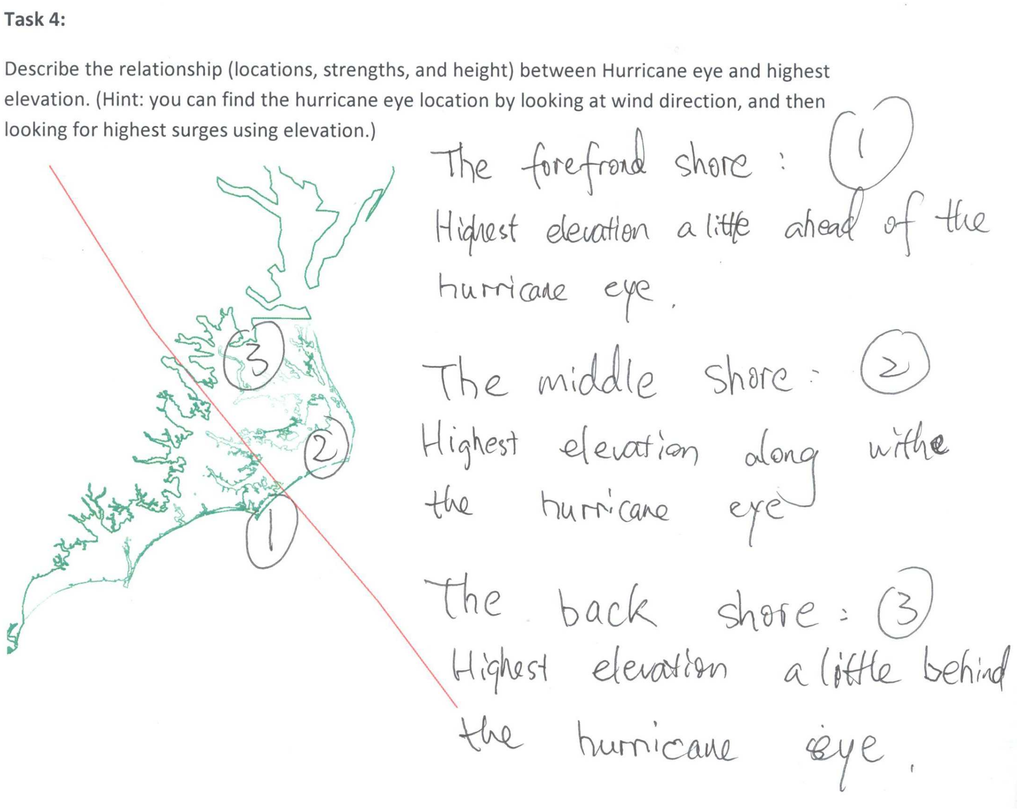
Ending : 320

Figure 49: Sample answers from subjects for task 3. Participants marked back surge area and wrote down the time duration.

North Carolina coast of the entire time duration displaying elevations; the second phase used a focused view around back surge area detected by our algorithm during the storm surge time displaying velocity vectors along with elevation surfaces; and last phase showed the elevation changes after the hurricane passed.

Task 4: Describe the relationships (location, strength, and height) between the hurricane eye and the highest elevation.

Subjects were asked to find the relationships between wind vectors and water elevations. The wind vectors clearly identified the hurricane eye and eye wall and showed how it affected the elevations to change. A sample answer from subject of this task is shown in Figure 50. Some hints including the temporal changes of the major feature were provided for the subjects to understand the questions. In this case, the temporal curves of elevation changes and atmospheric pressure were displayed. The interactive systems enabled rendering elevation surface and wind vectors.



Task 4:

Describe the relationship (locations, strengths, and height) between Hurricane eye and highest elevation. (Hint: you can find the hurricane eye location by looking at wind direction, and then looking for highest surges using elevation.)

The forefrond shore : 1
Highest elevation a little ahead of the hurricane eye .

The middle shore : 2
Highest elevation along withe the hurricane eye

The back shore : 3
Highest elevation a little behind the hurricane eye .

Figure 50: Sample answers from subjects for 4. The red line indicated the hurricane path. Participants wrote down the answers.

The animation used two phases to describe this event by rendering the elevation and wind direction using a focused view on North Carolina coast in the first phase and a dynamically view following the hurricane eye to observe the wind in the second phase. Both phases only animated time durations when hurricane was close to North Carolina.

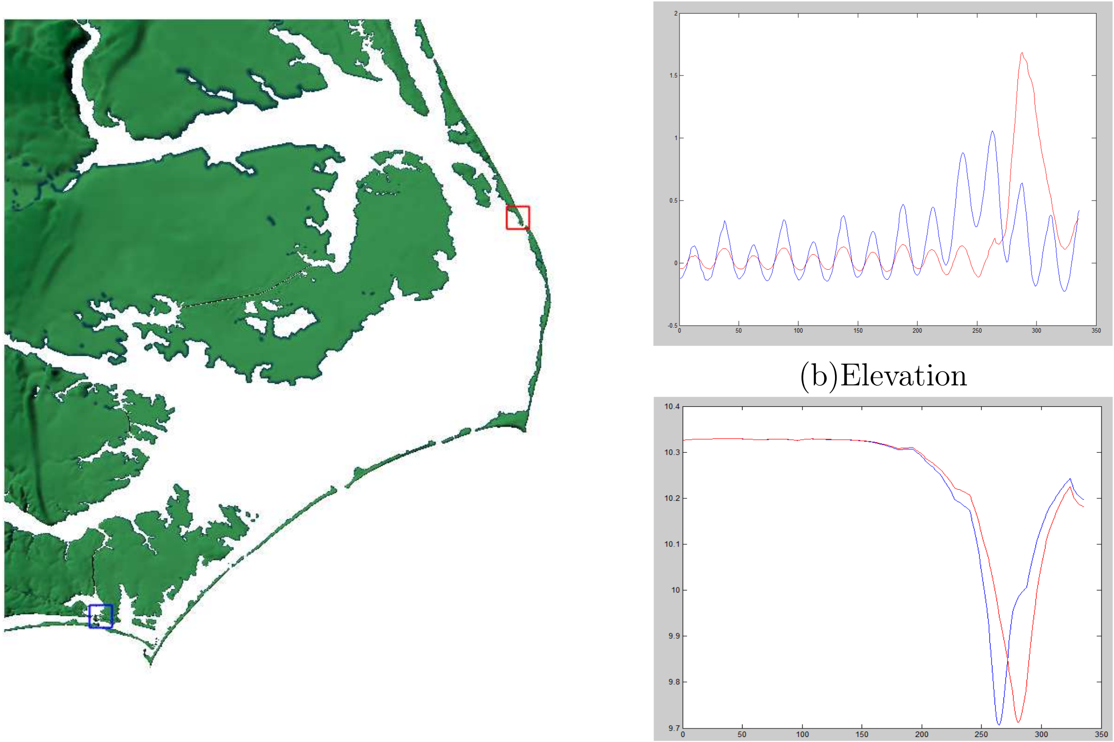
7.2.6.3 Category 3 - Reasoning

In this category, the subjects would look at very low level details on certain locations in the simulation. We focused on if and how fast the subjects could find out the reason of different behavior of two locations during the hurricane. The overall changes of relevant variables in the detailed area would also be shown to the subjects.

Task 5: The two locations were NOAA water level observation stations on Outer Banks. One is Oregon Marina Inlet, the other is Beaufort. The vertices around these two areas were extracted and the overall changes of these vertices are shown to the subjects. An overview of how hurricane traveled was presented to the subject and then the relevant variables during the storm surge time were displayed. The question is: Why the elevation and pressure changes are different from each other on those two locations?

Figure 51 showed the locations of observation stations on Outer Banks and the temporal trends of two different attributes. The animation displayed the focused view of these two locations rendering relevant variables with the highlighted locations. The animation displayed overall changes and then the changes during storm surge time period using three phases. The first phase displayed the overall changes over the entire time period. The second phase showed the elevation with velocity vectors to help subjects find the reason for the differences in the elevation curves. The last phase displayed the wind vectors with atmospheric pressure to help subject inspect the pressure changes.

Task 6: Two canals in North Carolina which might be dramatically affected by the storm surge were analyzed. One was the Pungo River Canal; the other was Adam Creek Canal. The two canals were impacted differently during the hurricane because of their locations to the hurricane path. The vertices around these two areas were extracted and the overall changes of these vertices are shown to the subjects. The subjects were asked to find out the reason why the two canals changed differently during the storm surge. The question was: Why does the Pungo River Canal and



(b)Elevation

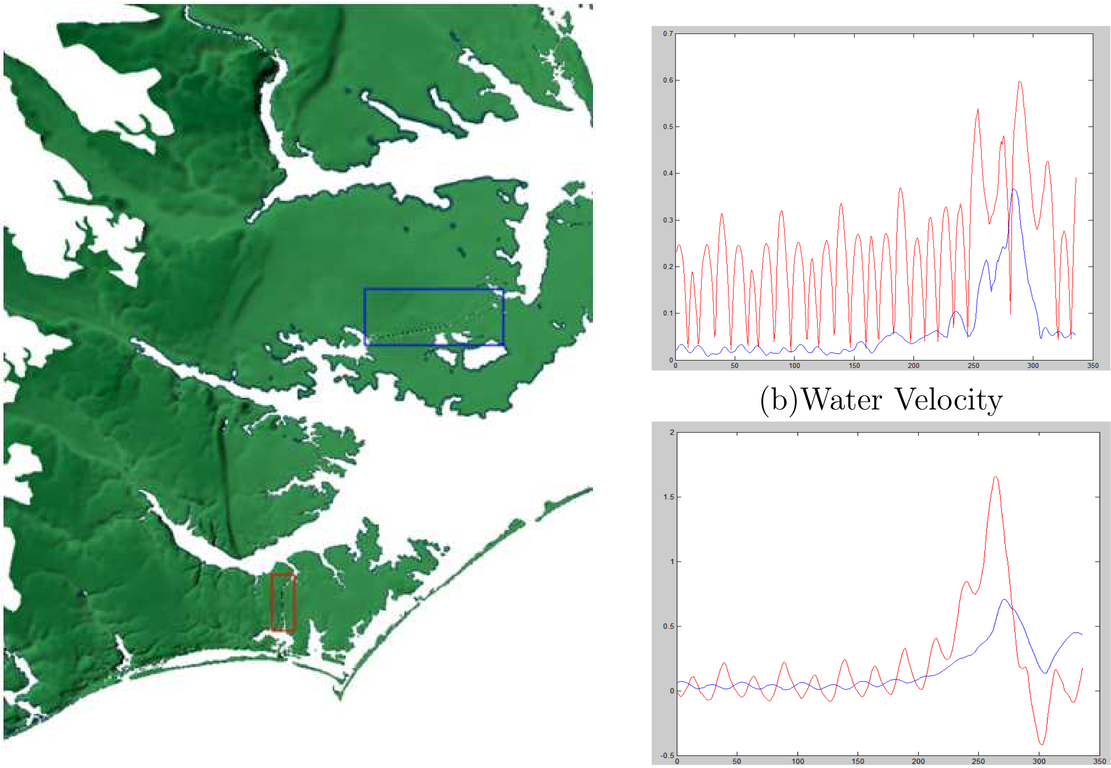
(a)Task 5, two locations on Outer Banks

(c)Atmospheric Pressure

Figure 51: (a) This image shows the two locations in task 5 on Outer Banks, marked with red and blue box. (b) and (c) show the temporal trends of elevation changes and atmospheric pressure of these two locations.

Adam Creek Canal have different elevation changes?

Figure 52 showed the locations of the two canals in North Carolina and the temporal trends of two different attributes. The animation showed an overview of changes during the entire time duration, rendering elevation surfaces, then displayed elevation and water velocity of each location with a closer view during the surge time duration one by one.



(b)Water Velocity

(a) Task 6, two canals in North Carolina

(c)Elevation

Figure 52: (a) This image shows the locations of the two canals in task 6, marked with red and blue box. (b) and (c) show the temporal trends of water velocities and elevation changes of these two locations.

7.2.7 Data Analysis and Results

7.2.7.1 Completion time

Table 3 shows the average completion time, standard deviation and t-test result of each task. Figure 53 shows the completion time and STD error bar of all the tasks. T-test results show significant differences in overall results and task 1, 4, 5, and 6. The T-test result of overall is 0.000005, and for task 1 to 6 are 0.002, 0.106, 0.106, 0.049, 0.041 and 0.009 respectively.

We analyzed the completion time of all tasks disregarding the correctness of the answers. Generally the subjects took less time to finish their tasks using animation

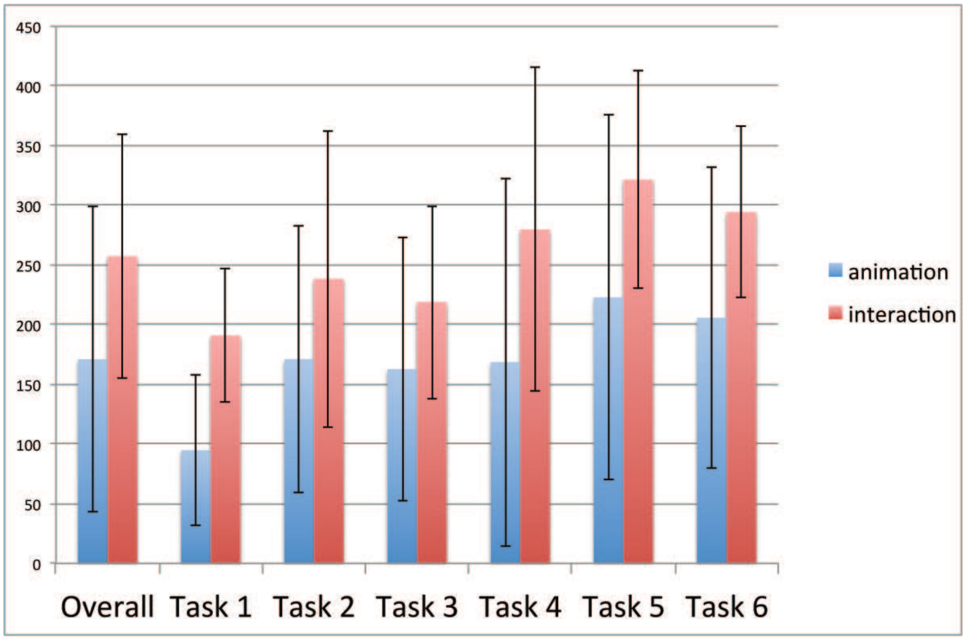


Figure 53: Task completion time and STD error bar. The first column is the overall average;The rest are tasks 1-6.

Table 3: Task completion time and STD error

Case	Animation	Interaction	t-test
all average	170.9861111	257.3611111	5.04182E-06
task 1 average	94.75	190.9166667	0.002455855
task 2 average	171.1666667	238.5	0.105595806
task 3 average	162.8333333	218.9166667	0.10584544
task 4 average	168.6666667	279.8333333	0.04891159
task 5 average	222.8333333	321.5833333	0.04069019
task 6 average	205.6666667	294.4166667	0.009488679

than interaction. We observed significant differences in completion time in task 1, 4, 5 and 6. This supports our hypothesis 1.

7.2.7.2 Accuracy

We first describe our grading of each task and then present the result of accuracy.

Task 1 & 2: Since the answers in this category are all drawn, we scan the answer papers and compare the answers with our ground truth. Pixel-wised differences are calculated.

For task 1, 20 points are evenly selected on the ground truth path. We then find the nearest points to each on the path subjects drew. The sum of the distance between all those points gives quantitative results of task 1.

For task 2, we compare the areas (number of pixels) from answer papers and the ground truth. The ground truth and sample answer image is shown in Figure 48.

Task 3 & 4: The grading for task 3 is straightforward. Each of the answers on the staring time step, the ending time step, and location, credits 33%. We allow a +/-5 time step differences in answering time step values.

The answers for task 4 is that there are several different phases of the relationship between highest elevation and hurricane, such as highest elevation happened ahead of or behind to the east of hurricane eye along the path. We give corresponding portion of the grades to each correct description.

Task 5 & 6: Task 5 and 6 focus on letting the subject tell the reason of why different locations have different impacts. The correct answers are the different distances between locations and hurricane eye, as well as the time when hurricane passed. Credits are given if the subjects described the reason perfectly or partially mentioning the key words.

Table 4 shows the average accuracy, standard deviation and t-test result of each task. Figure 54 shows the accuracy and STD error bar of all the tasks. T-test results shows no significant differences in the results.

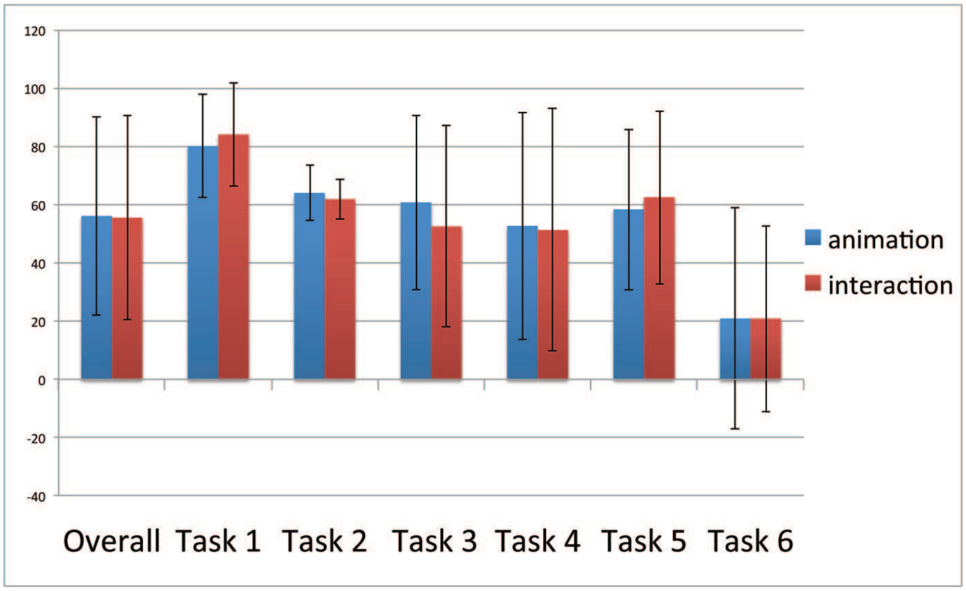


Figure 54: Task accuracy and STD error bar. The first column is overall average of all the tasks; The t-test of all tasks doesn't show significant differences.

Table 4: Task accuracy time and STD error

Case	Animation	Interaction	t-test
all average	56.10363889	55.50241667	0.432729
task 1 average	80.066	84.0745	0.338127
task 2 average	63.9725	61.85666667	0.302523
task 3 average	60.75	52.5	0.247851
task 4 average	52.66666667	51.25	0.396065
task 5 average	58.33333333	62.5	0.337119
task 6 average	20.83333333	20.83333333	0.5

7.2.8 Discussion

From the results, we can observe that subjects usually take less time to complete the tasks using animated visualization, which supports our hypnosis 1.

The overall accuracy is low in this study. Both animation and interaction are averaging around 55%, indicating some tasks are more difficult than we anticipated. Though the average accuracy of animation is slightly higher than interaction, we does not observe significant differences in the results. This indicates our hypnosis 3 is not supported by the results.In some case, the interaction achieves a better accuracy than animation. The reason is that subjects gain more time to inspect one specified static time step and perceive more details, for example, task 2. The presentation category takes the least time to complete and highest accuracy. This finding supports our hypnosis 2.

We consider the last category more difficult than the other two and probably will take more time and be less accurate. The reasoning category does not reach a high accuracy; probably due to that computer science students are not familiar with the storm surge effects on elevation and land. Both interaction and animation receive a low accuracy, but animation still out performs interaction on task completion time. We plan to perform user studies on Meteorology students and faculties to find if they can do better than computer science students.

Another interesting aspect of our analysis results is that since the subjects fell into two conditions: animation first or interaction first. We can observe differences between these two conditions. For interaction first methods, the average accuracy of is 49.23% while later the animation method achieve 66%. This is reasonable because the later method would have a better accuracy because the subjects are getting familiar with the visualization when using the first condition of the study. For animation first methods, the average accuracy of animation is 59.48%, as the interaction accuracy is 61.77%. As we can see, comparing the first used method and the latter used method,animation always achieves a better accuracy than the interactive visualization. Also, the improvement from both conditions also shows animation performs better in perceive knowledges.

The average completion time of each task for Isabel is 3 minutes and 31 seconds while for Irene is 3 minutes and 39 seconds. The average accuracy of tasks using Isabel is 52.87% and for Irene is 58.74%. T-test result of accuracy for these two dataset is 0.1508 which shows no significant difference. This indicates our results are not affected by using two different datasets.

The average completion time of category 1, 2 and 3 is 176, 207 and 262 seconds respectively. The average accuracy for category 1, 2 and 3 is 72.49%, 54.29% and 40.63%. These results are consistent with the difficulties of each category.

We observed how subjects interact with animation. Most of them will choose to stop at one time which is related to the task and inspect that time step as they do using the interaction system. It is understandable static renderings can give participants more time to gain knowledges from the visualization. when users see interesting features or events during the animation, a stop function will be beneficial to inspect more details from the visualization.

In general, people perceive knowledge faster using animation. In our completion time analysis, animation totally outperformed the interactive visualization. Furthermore, the accuracy of both methods is close to each other which means necessary information can be observed and perceived from both visualization. Under the animation first condition, the subjects performs better in the following interaction methods too. The animation helps the subjects to better understand what is happening in the simulation. Because the subjects are from computer science department, they only have some common knowledge of hurricane behaviors. They can only conclude what they saw in the visualization. The results of accuracy shows that the animation is slightly better or equivalent to interactive systems.

7.3 Conclusion

In this chapter, we evaluate the effectiveness of animation in visualizing scientific time-varying data by conducting a formal user study. Our study compares two visualization systems. One system is our animated visualization, which has created the animations based on the focuses of tasks. The other is an interactive visualization system, which adopts a popular design with an interactive rendering panel and a 2D time curve panel. The study consists of tasks from three different categories, each of which serves a specific goal in visualization and visual analysis.

In general, the study shows animation is beneficial in some cases but interactive visualization also has its own strength too. The results show that animation is efficient but there are no significant differences in accuracy. This hints that user can achieve the same goal using animation faster than interactive visualization. We also observe participants behaviors during the user study. In some case, such as task 1 and 2, based on tradition and personal habits, interactive visualization is still users' first choice. In the representation category, the subjects are required to find the accurate position and areas in the map, the static display provides them the best results when they are recording their answers to the sheet. Also when under interactive visualization, participants tend to select time steps and display them fast to achieve dynamic effects which simulates the animation to observe changes between time steps. This means switching between the views can have a beneficial effect. No matter dynamic or static, visualization always needs to fit the the purposes the task.

CHAPTER 8: VISUALIZATION OF PROTOCOL

8.1 Introduction

In this chapter, we visualize non-traditional time-varying data, the network security protocols, which can be considered as different behaviors at different time stamps, and use animated visualization to help in teaching information assurance classes.

Information assurance education for both college students and the general public has been well recognized by many universities as an important topic since the early nineties. For example, Pothamsetty has investigated 25 security courses offered by multiple universities that are designated as NSA Centers of Academic Excellence and found that most of them adopted a curriculum structure of introductory-advanced information assurance courses [106]. Our experiences in teaching introductory and advanced security courses have led us to realize that there exists a gap between the teaching of security primitives and protocols, which may severely impact the learning outcomes of information assurance education. It has been shown that a group of secure primitives may finally compose vulnerable protocols if they are inappropriately organized. Therefore, special efforts must be made in the course plan to cultivate the capability of students to select suitable primitives and organize them appropriately. We believe that an interactive education environment for demonstration and exercises can help bridge this teaching gap.

The objective of this project is to develop an innovative digital construction set by integrating the achievements in security education and visualization. We also design instructional demonstrations and hands-on experiments using the set to assist students to bridge security primitives and protocols. Our approach applies the pedagogical methods that have been learned from the success of children and adult education using electronic blocks or construction sets [146, 154]. Specifically, we treat security primitives as Lego pieces and protocols as construction results. Our work can serve two tightly integrated purposes: automatic demonstrations of protocol decomposition to help students understand the relationships among primitives and protocols, and hands-on experiments to cultivate their capabilities to manipulate primitives and design protocols that satisfy different security requirements. The latter is one of the ultimate objectives of information assurance education.

The main contribution of this research is a 3D digital Lego approach that visualizes security protocols effectively and automatically to teach information assurance courses. Compared to traditional methods, this approach attempts to better reveal the relationships among security primitives and protocols, thereby improving security education outcomes. Our design of 3D digital Legos allows other instructors to develop, share, and modify the sample Lego sets so that they can generate their own demonstration and experiment materials easily. Based on this approach, we have developed a prototype system with several important interaction functions that can be used as a user-friendly demonstration and experiment environment. We have also

[1] Lego is a trademark of the LEGO Group. Here we use it only to represent the pieces of a construction set.

performed initial evaluations to assess this Lego-based approach on teaching introductory security courses and received positive feedback.

This chapter is organized as follows. Section 8.2 discusses related work on construction sets in education and graphical approaches for information assurance courses. In Section 8.3, we present our efforts to explore suitable representations of security protocols using real Lego toys, which help us design the 3D digital Legos. Section 8.4 describes our approach to 3D digital Lego generation for visualizing security protocols. Section 8.5 presents the integrated system as a user-friendly demonstration and experiment environment. We describe our evaluation processes and results in section 8.6 and provide a discussion on our approach in Section 8.7. Finally, Section 8.8 discusses future extensions and concludes the chapter. This work is published in [157].

8.2 Related Work

8.2.1 Construction Sets

Construction sets have a venerable place in the history of education. Records show that as early as in 1800 appeared a building set for castles and walled towns [68]. In America, building blocks have been recommended to parents since 1826 [40].

Recently, the educational role of construction sets has been enhanced by the integration of computational media. For example, building blocks with sensors and fiber optic output were used to construct a speech-enabled alphabet set [48] or 3D structures for communicating to a computer [87]. Particularly, construction sets have been widely used in undergraduate robotics education. For example, Lego bricks [110] were used as the controllers for large Lego sets. The sets provided a wide space for students to make hypotheses about how things work and validate their assumptions [147]. Similar digital manipulations have been used in artificial intelligence, programming, and general engineering courses [97, 99, 146]. Inspired by the success in robotics education, digital construction sets have been applied to the design of space habitat and vehicle [72] and computer systems [38]. For example, the functional decomposition approach [38] has been applied to many systems, including analog electronics, digital design, VLSI, and software.

In this chapter, we present an approach that adopts the concept of Legos to help students understand the relationships among security protocols and the involving primitives. Different from previous methods, our approach can automatically generate specialized digital Legos for various security protocols.

8.2.2 Achievements in Security Education

This project is inspired by the fact that various security protocols are constructed by a limited number of primitives. For example, Millen *et al.* [100] have summarized ten reduction rules to decompose security protocols into simple units and Cremers [39] has investigated how to decompose a complicated protocol into sub-protocols. Therefore, we believe that a suitable design of digital Legos can be used to assist us in teaching security protocols. Previously, we have developed a 2D Lego system for security courses [143] in which special 2D Lego pieces are designed to visualize the operations such as encryption. Our experiences show that the 2D shapes may cause some difficulty in understanding the security protocols, since the message contents are represented as embedded boundaries. Therefore, in this chapter we present a more intuitive approach that can simulate the real 3D Legos.

Several other graphical approaches have been proposed for security education. For example, Burger and Rothermel presented a general purpose simulation architecture for teaching security protocols [28]. Saul and Hutchison developed a graphic environment for analyzing security protocols [116]. Schweitzer [117] designed an interactive visualization tool for demonstrating protocols visually in a user-controlled stepwise manner. Elmqvist also developed an animation function to display protocols in a step-by-step fashion [49]. In contrast to our approach, these methods are mainly designed to use graphics or interactions to emphasize the sequential events associated with a protocol. Compared to previous methods, our approach can illustrate the messages of a protocol in visual forms and demonstrate the relationships among primitives and protocols.

8.3 Construction with Real Legos

8.3.1 Notation

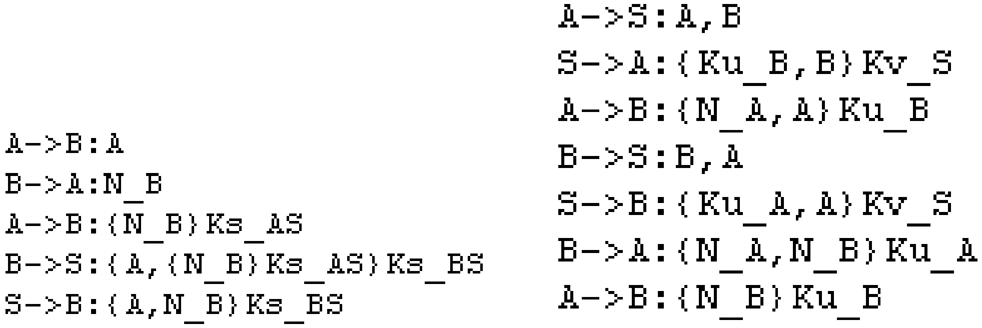
We first introduce the notations that are used to describe a security protocol in the remainder of this chapter. A security protocol usually consists of the interactions among multiple entities. We adopt the Dolev-Yao model [45] to represent the deduction capabilities of the legitimate entities and attackers. Table 5 lists the notations of the security protocols.

To build a generic approach that can represent a wide range of security protocols and attacks, we have adopted a flexible two-tier construction method [143].

We use the subindex of an item to label its owner so that the end users of our

representation	meaning
A	entity A
N_B	random number generated by B
K_u_B	the public key of entity B
K_v_B	the private key of entity B
K_s_AB	the symmetric key shared between A and B
$A \rightarrow B$	A sends a message to B
x, y	concatenation of items x and y
$\{msg\}key$	a message encrypted with the key

Table 5: Notations of the symbols.



```
A->B:A
B->A:N_B
A->B:{N_B}Ks_AS
B->S:{A,{N_B}Ks_AS}Ks_BS
S->B:{A,N_B}Ks_BS
```

```
A->S:A,B
S->A:{Ku_B,B}Kv_S
A->B:{N_A,A}Ku_B
B->S:B,A
S->B:{Ku_A,A}Kv_S
B->A:{N_A,N_B}Ku_A
A->B:{N_B}Ku_B
```

Figure 55: Two example security protocols, Woo Lam Protocol (left) and PKP protocol (right), shown in plain text.

system can directly edit the protocol files. For example, N_B represents a random number generated by entity B. If the subindex contains two entity names, it is shared between them. For example, K_s_AB represents a secret key shared between entity A and entity B. As illustrated in Figure 55, this approach represents messages of security protocols with plain text and they can be easily understood by the end users.

8.3.2 Protocol Construction with Real Legos

Before designing 3D digital Legos, we have explored several ways to use real Lego blocks to construct security protocols. This experiment helps us learn how the concepts of Legos can be used to represent primitives and protocols and assists us in designing effective 3D digital Legos for security education. The results also confirm our hypothesis that Legos can be used as an appropriate metaphor in an education tool to expose the relationships among security primitives and protocols. Below we describe our selections of protocol representation and the designs of protocol construction.

For constructing various protocols using real Legos, we have selected a Lego product that satisfies two requirements. First, we look for products that contain small Lego pieces so that the final construction results are in an appropriate size for demonstration and storage. Second, we need a large number of Lego blocks with similar shapes, since primitives usually appear multiple times in a protocol. Under these two requirements, we have selected the "Lego System Ultimate Building Set" made by LEGO as our tool.

We have explored several ways to construct security protocols with real Legos. We use colors to differentiate entities. For example, in Figure 56, red and yellow, blue and white, or green and white are used to visualize entities A and B respectively. We choose one or several Lego blocks to represent the primitive types. To utilize the available Lego pieces efficiently, we select combinations of Lego shapes for different primitive types carefully through the following procedure. First, we summarize the frequencies of primitives in several security protocols that are taught in our introductory level security course. Then the number of each Lego shape is counted. By matching the numbers of available Lego blocks to the frequencies of primitives, we ensure that our design can utilize the available Lego blocks efficiently.

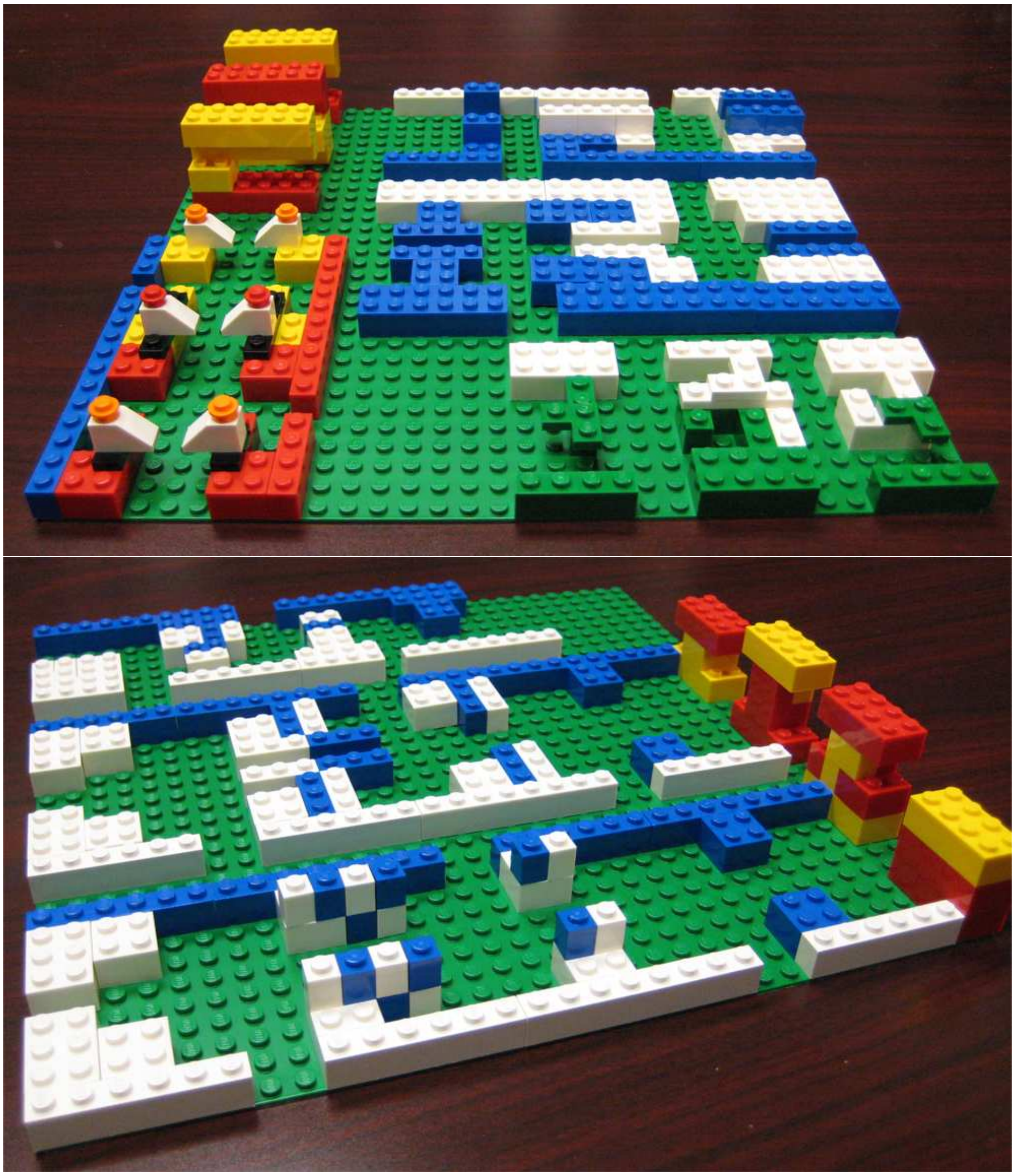


Figure 56: Example results of security protocols built with real Legos. Five designs are shown on the left and four are shown on the bottom

Based on the designs of primitives, we have explored several methods to construct protocols. Figure 56 shows five designs for the Needhand-Schroeder-Lowe protocol on the top and four designs for the Andrew Secure RPC protocol on the bottom. Our main choices are between the vertical and flat designs for the message contents. For example, the top left red-yellow design in Figure 56 (top) is a vertical version for providing a strong transition impression and the blue-white designs in Figure 56 (top) and (bottom) are flat versions for demonstrating message contents. It is interesting to note that multiple ways can be used to construct a protocol even with a simple Lego set. Also, this experiment helped the authors to remember several security protocols easily.

8.4 Automatic Construction of 3D Digital Legos

We design a method to construct specialized 3D digital Legos automatically for teaching security protocols. This method allows more flexible generation of instructional demonstrations and hands-on experiments than real Legos. Compared to the traditional text-based methods (examples shown in Figure 55), our Lego-based approach provides more effective course materials to direct the students' focus and attract their interests.

In this section, we present a generic method to construct 3D digital Lego sets for teaching various security protocols. Our method is developed based on the two-tier protocol representation that enables our approach to visualize different security protocols and attacks. The entire generation process is automated to allow easy creation and sharing of course materials.

8.4.1 Basic Lego Design

To better expose the relationships among primitives and protocols, we use different shapes to represent the primitive types and different colors to represent the entities. For each Lego block, only one surface is chosen to carry the information of message contents and is used to determine whether or not two blocks can fit together. In this way, a protocol can be visualized as multiple sending and receiving blocks.

Specifically, our digital Legos are constructed with the following procedure. First, we generate a set of geometry meshes to represent the primitive pieces based on 2D designs. Second, multiple blocks of digital Legos are composed in an appropriate order to visualize a security protocol.

Since we want to construct the digital Lego blocks automatically for a given protocol, we use two portions with fixed shapes and two portions with adjustable shapes to compose one Lego block. As shown in Figure 57, the top, bottom, and body define the general shape of a Lego block and the content surface is generated according to the message content. The shapes of the top and bottom portions match each other to ensure the vertical connection between any two blocks. They always point downward, since we assume that the protocols are executed from top to bottom. The content surface carries the most important information so we use a later subsection to discuss its generation in detail. The length of a block is also automatically adjusted according to the content of a message.

Our main purpose for separating the sending and receiving blocks is to provide flexibility to the demonstration and experiment tasks. Although a message is shared

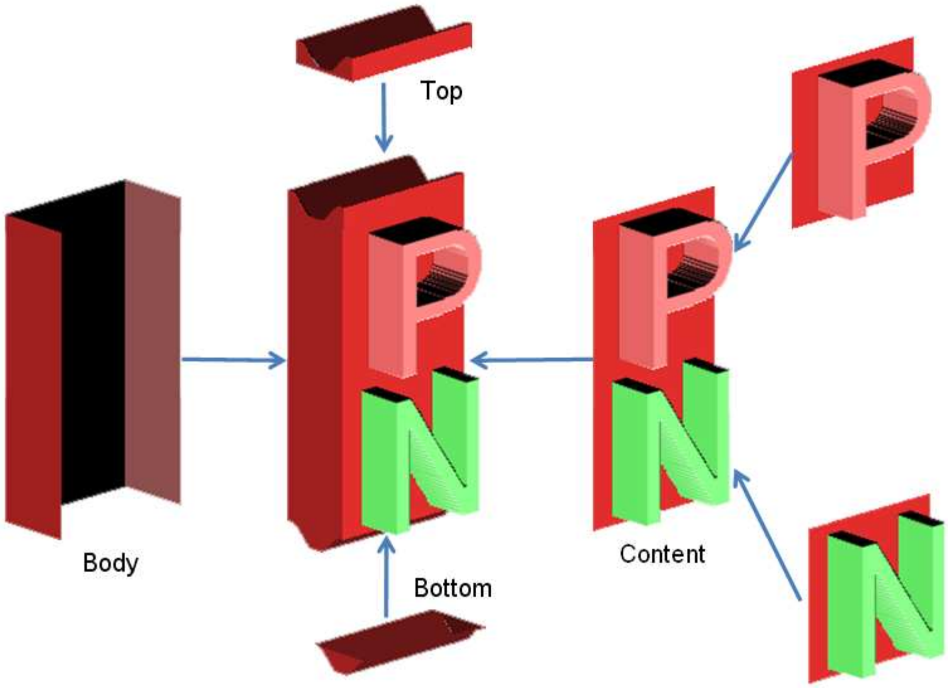


Figure 57: Our Lego block consists of two portions with fixed shapes (top and bottom) and two portions with adjustable shapes (body and content). The adjustable portions are automatically generated according to the message content and sending/receiving type.

between a sender and a receiver, their interpretation of the same message may be different, especially when attackers are involved. This also allows us to show different detail levels of the same message in demonstration and experiment tasks.

Once a protocol is selected, this Lego design allows us to generate all the Lego blocks automatically. For better discretion of the Lego body and the message contents, we use a similar, but deeper color for the Lego body. We believe that this design matches the spirit of Legos closely, which is to capture the interests of students and attract their focus to important security concepts represented by functional Lego shapes.

To generate 3D Legos, we use polygon meshes because of their flexibility. As shown in Figure 57, the top, body, and bottom portions are composed of simple 3D

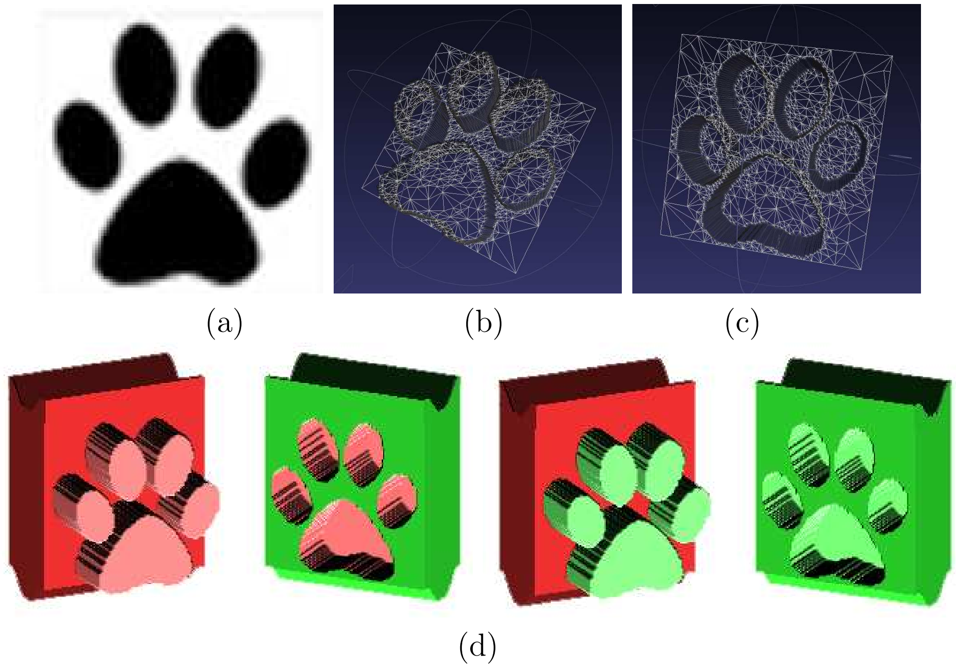


Figure 58: The generation process of primitive pieces. (a) We design grey-scale images to represent primitive shapes. (b) The input image is automatically converted to a 3D mesh to represent the sending operation. (c) We reverse the point heights in (b) to represent a receiving operation. (d) Sample results of using this primitive piece during Lego visualization.

meshes. We use the following two subsections to describe our procedures to generate the content surfaces.

8.4.2 Design and Generation of Primitive Representations

Since the message surfaces are rendered in 3D, we also prepare our primitive pieces in 3D so that they can be used to compose 3D Lego blocks quickly during the rendering process. The following describes our method that allows users to design the shapes of primitives by transforming 2D images to 3D meshes.

Our method allows users to design their primitives using grey-scale images, as shown in Figure 58 (a). Instructors can use any image editing software to input their design easily and the rest procedure is automatically handled by our method.

Specifically, we map a $n \times n$ (100 is used for all the examples in this chapter) grid on the input image, and preserve all the line connections in the grid. The pixel colors (darkness values) in the image are used to adjust the corresponding point heights in the grid. This procedure generates an initial 3D mesh that matches the appearance of the input image. We also use the point heights to separate the raised portions from the background and assign them to different colors during visualization. To improve the efficiency of the rendering process, we simplify these meshes with the MeshLab software [33] to generate the final primitive pieces. Figure 58 (b) shows the generated mesh based on image (a) and the protruding surface is used to represent a sending operation. We reverse the mesh in (b) to generate the receiving piece with a dented surface. This design ensures that two content surfaces can be put face-to-face if and only if their shapes match. The content surface is then combined with the other portions of a Lego block to generate the final results in Figure 58 (d).

We have designed several Lego sets to cover all the primitives in our selected uniform representation of protocols, as shown in Figure 59. These results demonstrate that our approach can generate various Lego sets flexibly. This method also allows other users to share these designs and create their own shapes easily. We believe that the ability to switch primitive designs can help users to choose their desired styles and make the learning process more attractive to students.

8.4.3 Generation of Content Surfaces

With the primitive pieces created above, we can automatically generate the content surfaces of Lego blocks for a given message. To compose a connected 3D mesh as

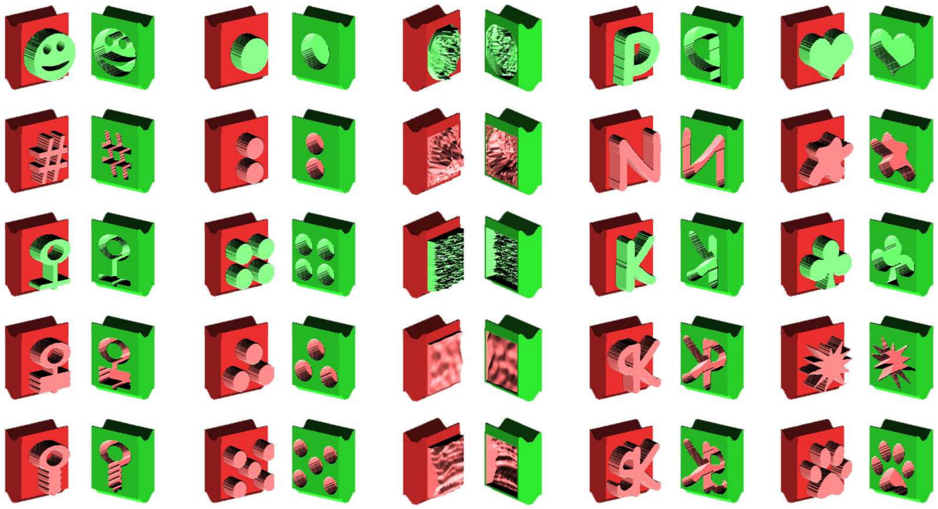


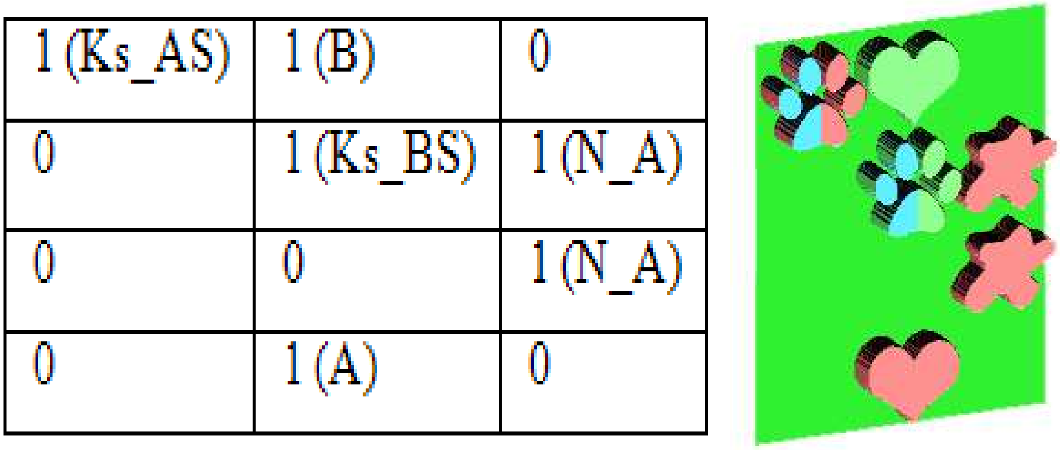
Figure 59: Examples of our primitive designs. From top to bottom are principal/entity, random number, public key, private key, and symmetric key. The pieces with red bodies are sending blocks and the green bodies are receiving blocks.

the content surface, we use the following procedure which first arranges a message content on a 2D table and then stitches corresponding primitive pieces together.

A message often consists of a list of primitives connected by manipulation operators such as encryption and concatenation.

We can view concatenation as the connection of two or more primitives at the same level, and encryption as the coverage of primitives at a deeper level. A 2D table can be generated for any given message. For example, Figure 60 (left) shows the filled 2D table for message "{B,{N_A,N_A}Ks_BS,A}Ks_AS" in the Yahalom protocol. Starting from a corner of the 2D table, we fill it with the message content by increasing the row when seeing concatenation or increasing the column when seeing encryption. We also record whether or not a location on the table has content or not by assigning a 0/1 flag to it. In this way, we can use such a 2D table to represent any message.

During the real-time rendering process, we draw Lego blocks according to their



1(Ks_AS)	1(B)	0
0	1(Ks_BS)	1(N_A)
0	0	1(N_A)
0	1(A)	0

Figure 60: The generation process of a content surface. A message "{B,{N_A,N_A}Ks_BS,A}Ks_AS" is first converted to a 2D table (left), then rendered with corresponding primitive pieces automatically (right).

content tables. For locations without any content (with flag 0), we draw one big polygon to cover the space. For locations with contents (with flag 1), we draw the corresponding primitive pieces in the pre-assigned colors of their entities. For symmetric keys, we take the colors of both entities and use each to draw half of the mesh. Figure 60 shows an example of our rendering method.

Here we use a security protocol, Woo and Lam Pi 3 protocol, as an example to illustrate our rendering process. Figure 61 shows the traditional text version of the protocol on the left, and the digital Lego version on the right. We use colors to represent the communicating entities in the protocol: red for Alice, green for Bob, and blue for the server. In this example, we choose the last style shown in Figure 59, with the heart shape representing an entity, the star representing a random number, the club representing a public key, the claw representing a symmetric key, and the echinus shape representing a private key. Each row of the rendering represents the transition of one message. For example, the first row is Alice sending her identity A

```
A->B:A
B->A:N_B
A->B:{N_B}Ks_AS
B->S:{A,{N_B}Ks_AS}Ks_BS
S->B:{A,N_B}Ks_BS
```

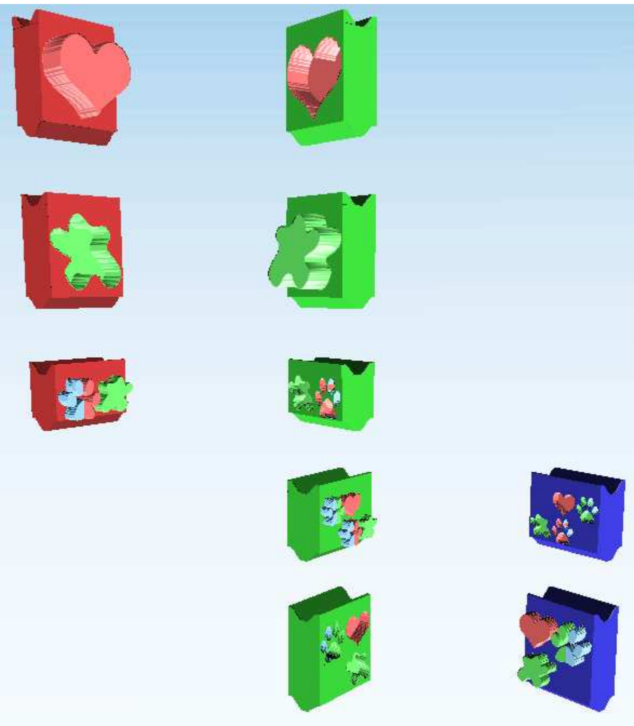


Figure 61: Woo and Lam Pi 3 protocol shown in the text-based approach (left) and the Lego-based approach (right).

to Bob. The convex and concave shapes are used to indicate the sending/receiving operations.

8.5 Integrated Lego System

With our digital Lego sets, we have developed two types of course materials: protocol demonstrations and hands-on experiments. The demonstrations are designed to better illustrate important protocol concepts during lectures. As a complementary component, our hands-on experiments are developed to train students to apply security knowledge flexibly during protocol design. We have integrated both components into one prototype Lego system, so that students can study examples and take exercises with the same tool.

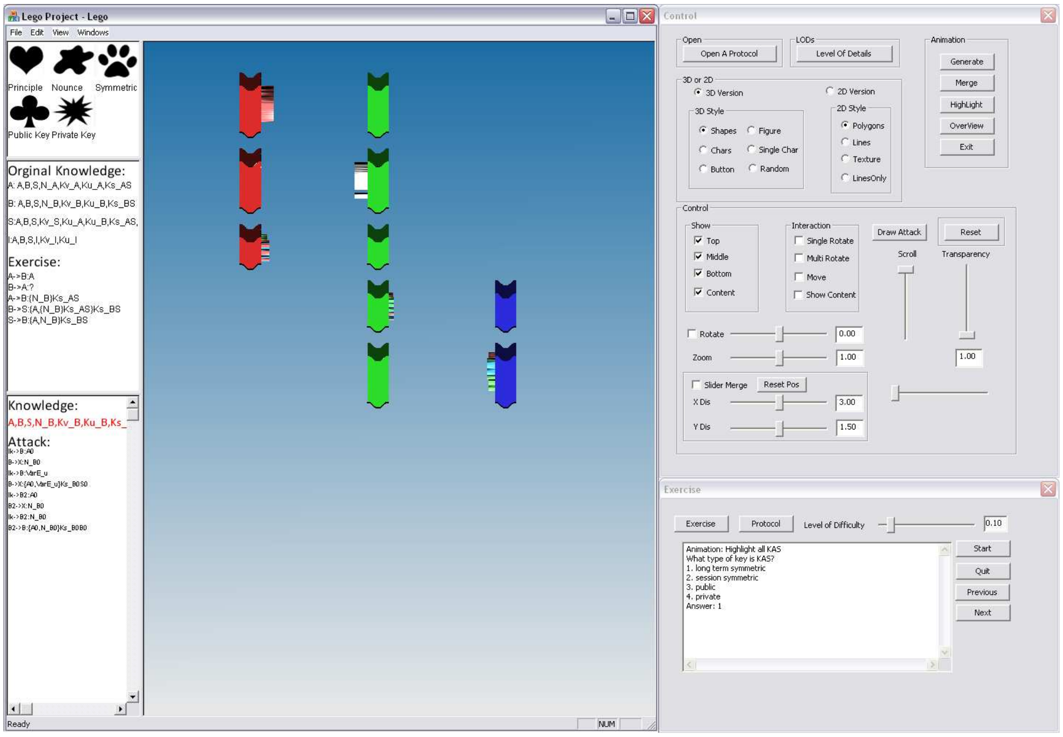
8.5.1 System Design

We develop our system with a multi-panel interface design. As shown in Figure 62, our system is composed of a main rendering window on the left and two interaction windows on the right. The main window contains four panels: primitive panel for displaying the current primitive design (left top), protocol panel for node knowledge, protocol contents or exercises (left middle), attack panel for attack strands and the knowledge of a node selected by users (left bottom), and rendering panel for visualizing and interacting with 3D Legos (right). The right top window is designed for users to adjust the rendering and interaction settings. The right bottom window is for controlling the exercise process. This multi-panel interface allows us to integrate multiple demonstration and experiment functions into our Lego system.

8.5.2 Interaction

Since we visualize security protocols with 3D digital Legos, it is important to provide suitable interactive methods that allow users to browse Lego contents freely and assist them in constructing protocols. As examples show in Figures 63 and 64, our system is capable of the following specialized interaction functions:

- Rotating: Viewers can rotate individual or a group of Lego blocks.
- Moving: Viewers can also move individual or a group of Lego blocks around the screen space.
- Displaying messages: The message contained in a selected Lego block is displayed in a floating window.
- Facing-to-viewer: We design a special facing-to-viewer rendering function, which



Orginal Knowledge:
A: A,B,S,N_A,Kv_A,Ku_A,Ks_AS
B: A,B,S,N_B,Kv_B,Ku_B,Ks_BS
S:A,B,S,Kv_S,Ku_A,Ku_B,Ks_AS,
I:A,B,S,I,Kv_I,Ku_I

Exercise:
A->B:A
B->A:?
A->B:{N_B}Ks_AS
B->S:{A,{N_B}Ks_AS}Ks_BS
S->B:{A,N_B}Ks_BS

Knowledge:
A,B,S,N_B,Kv_B,Ku_B,Ks_

Attack:
Ik->B:A0
B->X:N_B0
Ik->B:VarE_u
B->X:{A0,VarE_u}Ks_B0S0
Ik->B2:A0
B2->X:N_B0
Ik->B2:N_B0
B2->B:{A0,N_B0}Ks_B0B0

● 3D Version ○ 2D Version
● Shapes ○ Figure ○ Chars ○ Single Char ○ Button ○ Random
● Polygons ○ Lines ○ Texture ○ LinesOnly
☑ Top ☑ Middle ☑ Bottom ☑ Content
☐ Single Rotate ☐ Multi Rotate ☐ Move ☐ Show Content
☐ Rotate 0.00
Zoom 1.00
Transparency 1.00
☐ Slider Merge
X Dis 3.00
Y Dis 1.50
Level of Difficulty 0.10

Animation: Highlight all KAS
What type of key is KAS?
1. long term symmetric
2. session symmetric
3. public
4. private
Answer: 1

Figure 62: Our system interface includes both 3D Lego-based and text-based interaction panels.

turns the message content surfaces of selected Lego blocks to viewers while preserving the central positions of these Lego blocks, as shown in Figure 63.

- Merging: Viewers can adjust the distances between adjacent Lego blocks and finally merge all of them, just like playing with real Legos. An example of this merging function is shown in Figure 64.
- Labeling: We also allow viewers to select important primitives or protocol portions and adjust their rendering parameters to emphasize the important contents.

8.5.3 Experiments

We also design an experiment function for our digital Lego system so that it can be used for practice and homework. An experiment panel is provided with the supporting

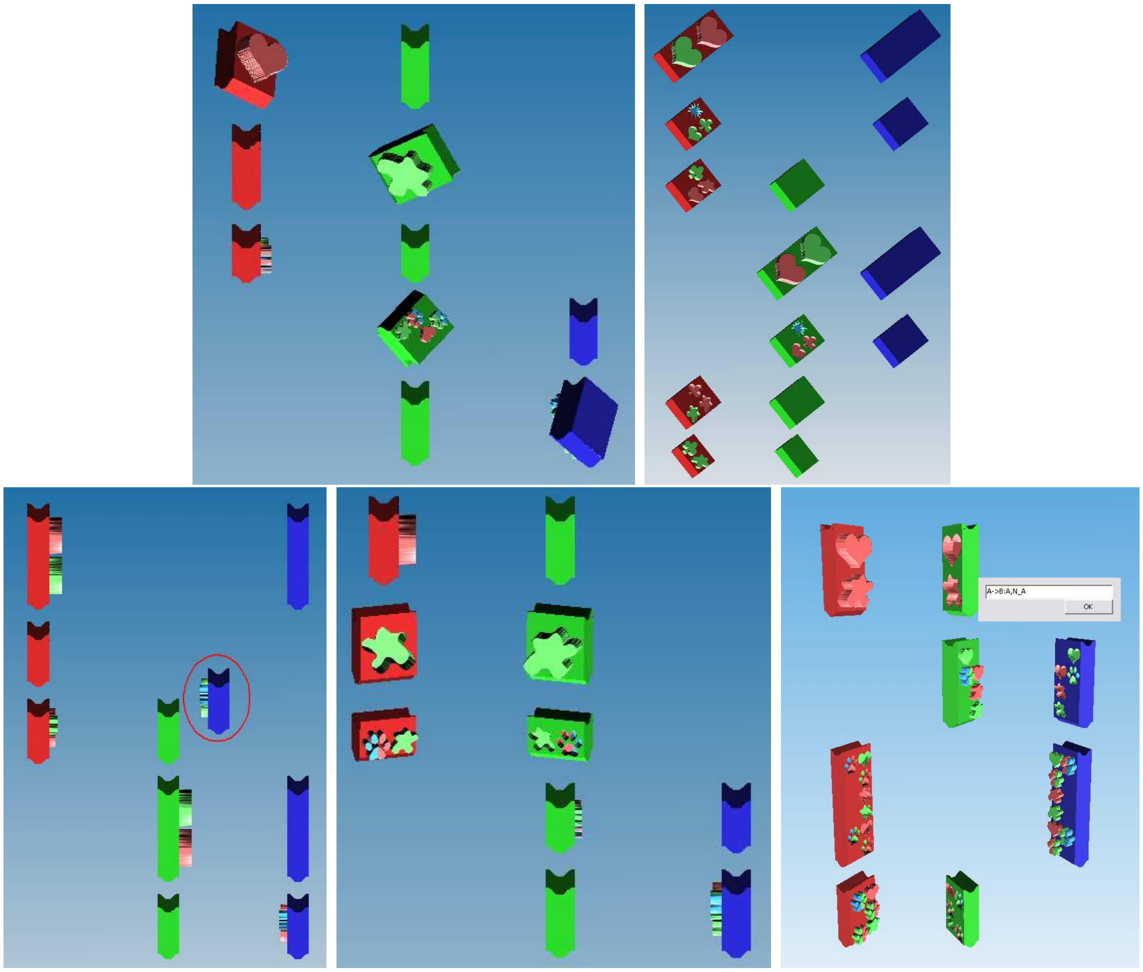

A->B:A,N_A

Figure 63: Interaction examples of the single rotation, multiple rotation, moving, facing-to-viewer, and displaying message functions.

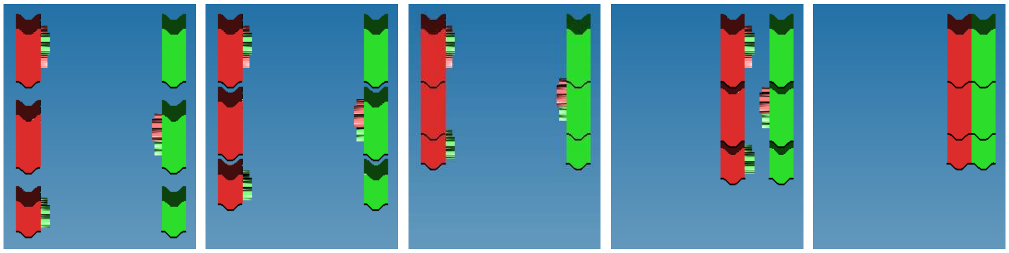


Figure 64: An example of the merging interaction. The Lego blocks in the same column are sent by the same entity and they are merged first. Then all the columns are merged.

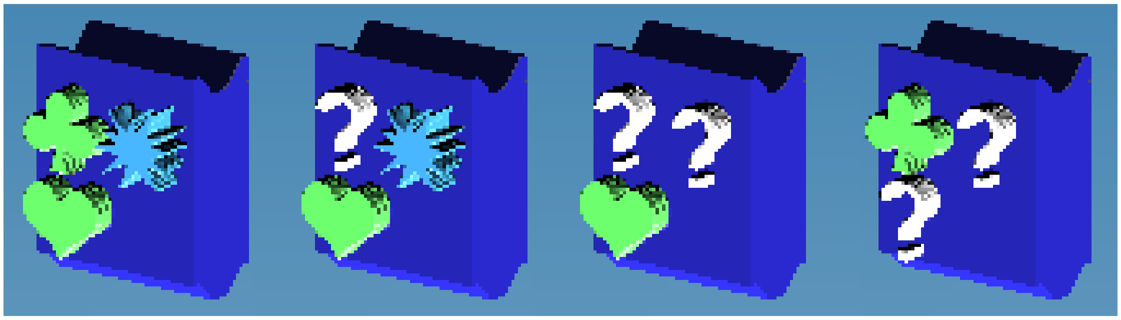


Figure 65: Experiment examples. Selected primitives can be automatically replaced with a "?" mark.

functions, such as start, next, and complete. Suitable rendering settings are also adopted during the visualization process. For example, Figure 65 (left) shows a Lego block in our filling experiment. This experiment is designed to emphasize important portions of protocols and strengthen related concepts by asking students to complete a pre-designed protocol. For each of our sample protocols used in the class, we randomly remove a portion of Lego blocks or messages that are related to the lecture contents. A difficulty level is used to control the amount of information that is hidden. As shown in Figure 65, our Lego construction method can automatically visualize the protocol. We can also change the settings of 3D Legos to direct the attention of users to specific portions of a protocol, such as using the facing-to-viewer motion in Figure 63.

8.5.4 Results

We have tested the Lego approach with all the security protocols being taught in our undergraduate course "Introduction to Information Security and Privacy". The selected protocols include the examples that are widely used in security courses, such as Woo Lam protocol, Neumann Stubblebine, Needham Schroeder Public Key, Needham Schroeder Lowe Public Key, and Otway Rees. We have also selected several protocols from real life applications such as BAN modified version of CCITT X.509 (3), Kerberos V5, and KSL (Nonce based improvement of Kerberos V5). Many of these protocols have been collected by the SPORE project [74]. For all these protocols, our approach can automatically generate the Lego-based protocol visualizations. Figure 66 shows the Lego representations of the following messages:

- "A->S:{N_B}Ku_B"
- "A->S:A"
- "A->S:{Ks_AB,N_B,A}Ks_BS"
- "S->A:{N_A,B,Ks_AB,{Ks_AB,A}Ks_BS}Ks_AS"
- "A->S:B,N_B,{A,N_A}Ks_BS"
- "A->S:A,N_A"
- "S->A:{B,Ks_AB,N_A,N_B}Ks_AS,{A,Ks_AB}Ks_BS"

Due to the space limit, in Figure 66 we illustrate messages with different lengths and complexity, instead of the entire protocols. Since a protocol is composed of individual messages, these examples demonstrate that our Lego construction approach can handle quite a variety of security protocols.

Figure 67 shows one message represented in several designs of security primitives. Other instructors can either adopt our samples directly or design their own appearances of primitives. Our approach to constructing digital Legos allows an easy switch of primitive designs during runtime.

We have also tested the usages of our interaction functions. When studying a protocol, we often use the merging function to visualize the entire protocol. Then,

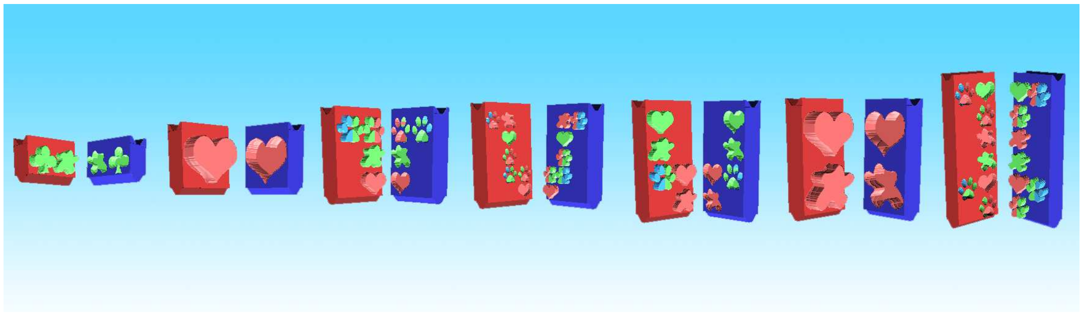


Figure 66: Example results demonstrate that our approach can visualize various security protocols.

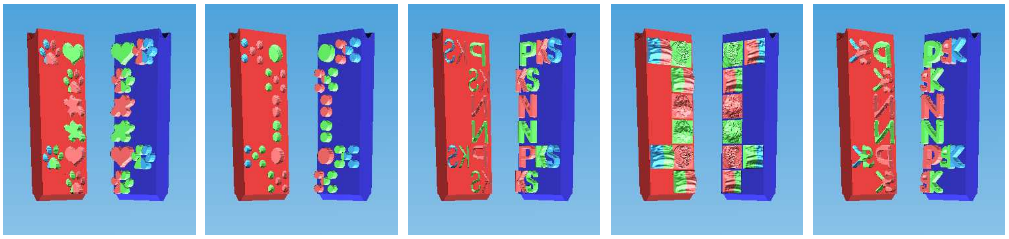


Figure 67: $S \rightarrow A : \{B, Ks\_AB, N\_A, N\_B\}Ks\_AS, \{A, Ks\_AB\}Ks\_BS$. The message is visualized in different primitive styles. Our Lego construction approach can switch among different primitive styles in real time.

we use the facing-to-viewer function to browse the contents of individual messages. For protocols with many rounds of interactions, we can use the moving function to scroll down the screen to view the entire protocol. For a particular Lego block, viewers can use the rotating function to observe details or use the labeling function to view the text representation of the message. We believe that these functions are essential to help instructors or students to experience realistic interaction with 3D Lego-represented security protocols.

8.6 Evaluation

We have designed and performed user studies to evaluate the effectiveness of our Lego-based approach on teaching security protocols. The main evaluation goal is to compare our Lego-based approach with the traditional text-based approach from different educational aspects. The results of these studies have provided important information to us on the advantages of the new Lego-based approach, as well as useful clues to improve this visual-based scheme.

Our evaluation plan consists of the following two portions: an informal survey for gathering feedback on the general Lego-based approach and a formal user study for assessing the specific performance of the Lego-based and text-based approaches. The following first describes the informal survey, which shows significant interests in using Legos in class from students. Later, we present two experiments in our user study and discuss their results.

8.6.1 Survey

Due to the limit of available time in our class, we have designed a brief survey to assess the general interest of students for Lego-based approaches. Our hypothesis is that an interactive tool based on a popular toy concept would pique the interest of students in computing majors more so than traditional text-based methods. We believe the positive results indicate that visual approaches can better promote students to study challenging and abstract security theories. The following lists the subjects, procedure, results and discussions of our survey.

8.6.1.1 Subjects

Our subjects include 23 student volunteers from the "Introduction to Information Security and Privacy" class at UNC Charlotte. The majority of our subjects are juniors with computing backgrounds. Since these students have learned security protocols using the text-based approach throughout the semester, they are all equipped with basic knowledge of security protocols and are familiar with the text-based approach.

8.6.1.2 Procedure

Before the survey, a fifteen minute introduction of our Lego-based approach is given to the subjects. Since these students are familiar with the text-based approach, we concentrate on explaining how the designs of digital Legos can be used to teach the primitive and protocol relationships in general security protocols. We also demonstrate the digital Lego system and its interaction methods. After the introduction, we answer questions raised by the students for about ten minutes.

During the survey, each student is given a copy of the survey questions and instructed to take as much time as they need to finish. The survey is in the form of multiple choice, Likert-scale, and free response questions. The questions are used to assess the interest of students on general Lego related issues. Figure 68 shows the four multiple choice questions and Table 6 shows the four Likert-scale questions (the six scales are strongly agree, agree, slightly agree, slightly disagree, disagree, and strongly disagree).

8.6.1.3 Results and Discussions:

The results of this survey indicate a strong motivation of students to combine text and visual based approaches to learn security protocols. Considering that the students participating in this survey have only been introduced to the Lego-based approach shortly, we think that they may have questions and concerns on the details of digital Legos. Even so, a majority of the students still choose the Lego-based approach in

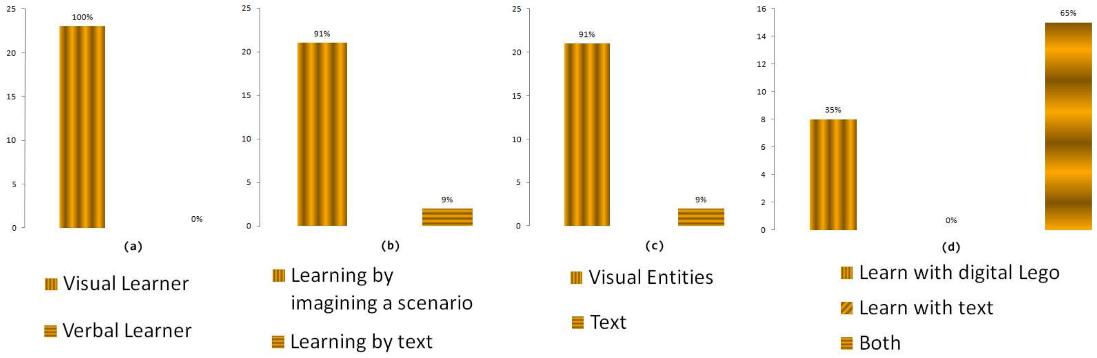


Figure 68: Multiple choice questions in the survey and their results shown in the bar graph. The questions are: (a) “Do you consider yourself as a visual learner or a verbal learner?”; (b) “If I were learning about a protocol, I would prefer to walk through each step by imagining a scenario where two or more entities execute the steps or use the text-based methods used in class so far?”; (c) “When I think of a security protocol, I get something most like visual of entities or text of a protocol?” (d) “I feel I can learn best by using digital Legos, text or both?” The colors represent the choices of students.

Questions	Strongly Agree	Agree	Slightly Agree
(a)	9	12	2
(b)	3	8	9
(c)	10	4	2
(d)	0	5	11
Questions	Slightly Disagree	Disagree	Strongly Disagree
(a)	0	0	0
(b)	0	3	0
(c)	3	2	2
(d)	3	4	0

Table 6: The results of the Likert-scale questions in the survey.
The questions are: (a) “I feel I can learn security protocols by a visual approach;” (b) “I feel I can learn security protocols by a digital Lego system;” (c) “In the past, I played with Legos a lot;” (d) “I feel I can learn security protocols by the text-based approach.” The numbers represent the choices of students.

additional to the traditional text based approach.

As shown in Figure 68, all the students consider themselves to be visual learners, which shows a unanimous interest in improving the traditional text-based approach. About 91% students prefer to learn security protocols as visual entities and imagine protocol scenarios in visual forms. This number indicates a wide acceptance of visual-based education tools. Also, in the last question "I feel I can learn best by digital Lego, text, or both?", none of the students chose the text-based approach and about 65% chose a combined Lego and text-based approach. Since our system is able to show the plain text of protocols as well as digital Legos, our system design matches the interests of students.

Table 6 shows our Likert-scale questions and their results. If we use the scores 0 to 5 to represent the choices "Strongly Disagree" to "Strongly Agree", the averages are 4.3, 3.35, 3.45, and 2.75 for questions (a)-(d), respectively. For the first question, "I feel I can learn protocols by a visual approach", the average score 4.3 shows a strong confidence in visual-based approaches. For the last question, "I feel I can learn security protocols by a text-based approach", the average score 2.75 is just a little bit higher than neutral. Since these students have been taught with the text-based approach, we think that this score indicates some obvious obstacles they have experienced during the semester. These two numbers match the results in Figure 68 as well. For sub-questions (b) and (c), we can see high percentages of students who have played with Lego in the past and who feel that they can learn security protocols by a digital Lego system. We find that four out of the seven students who have not played with Legos are also interested in the visual-aspect of this approach. This result indicates that the combination of digital Legos and text in our system may best serve for the purpose.

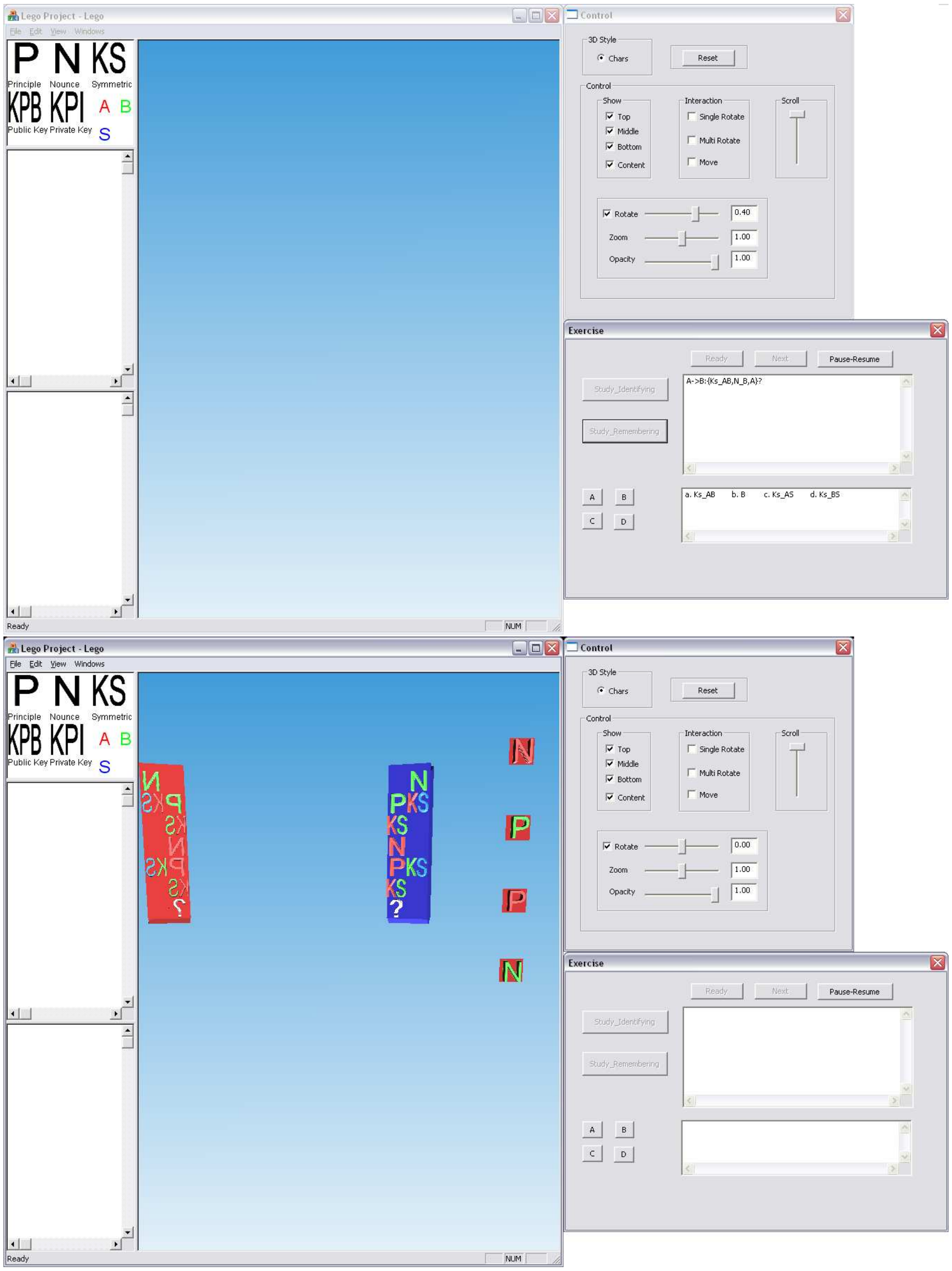
8.6.2 User Studies

The survey results support our contention that visual-based approaches should be used to improve the teaching of security protocols. We have further designed and performed two user studies to assess our Lego-based approach. Specifically, we concentrate on two essential aspects of learning: identification and memorization of security primitives and protocols.

We modify our digital Lego system to generate an experiment environment for our user studies. The following describes the three major changes.

- Adding automatic experiment functions, including randomizing question sequences, recording individual operation time durations and user answers, allowing pause and resume during the experiment, and saving subject files;
- Enabling and disabling experiment buttons for different experiment phases. During the observation phase, only one "question" button is active for viewing questions; during the response phase, only the multiple choice buttons are active; and the "next" button becomes active only after an answer has been selected. This function guides the subjects to finish the experiment without distraction.
- Adjusting the user control panels by hiding all unnecessary interaction buttons.

Figure 69 shows the interface of our experiment environment using the Lego-based approach and the text-based approach, respectively. The control buttons used during the studies are the same for both methods, so that they do not affect the study results.


A->B:{Ks_AB,N_B,A}?

a. Ks_AB b. B c. Ks_AS d. Ks_BS

☑ Rotate 0.40
Zoom 1.00
Opacity 1.00

☑ Rotate 0.00
Zoom 1.00
Opacity 1.00

Figure 69: The interface of our experiment environment. The top shows a question "$A \rightarrow B : \{Ks\_AB, N\_B, A\}$?" using the text-based approach with the questions and choices displayed in the right bottom panel. The bottom shows the same question using the Lego-based approach with the questions and choices shown in the middle 3D Lego panel.

Before the experiments, we hold a practice session to familiarize the subjects with our experiment environment and procedure. The procedure of the practice session is the same as our user studies, except that the practice session only contains one sample question and explanation for each experiment. This practice session is designed to reduce the confusion of subjects during experiments and ensure the accuracy of our captured time durations.

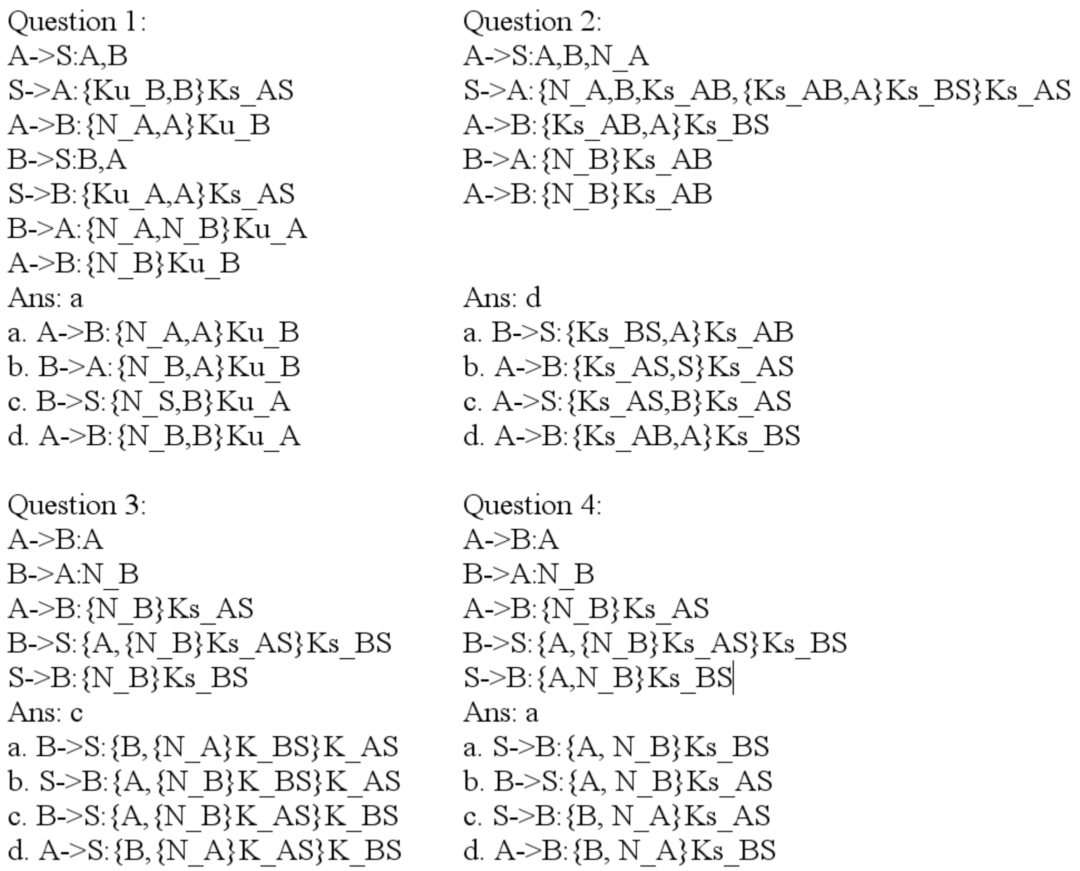
8.6.2.1 Experiment 1: Protocol Primitive Identification

Since the survey results have shown that the Lego-based approach can attract the attention of students, we are interested in finding out how this approach can assist the teaching of security protocols. Our first hypothesis is that the Lego-based approach could help students identify important primitives in a protocol more easily than the text-based approach. We design this experiment to evaluate the aspect of identification through measuring the factors of accuracy and time duration during identification tasks.

Apparatus: A Windows machine with an ordinary USB mouse.

Subjects: Seventeen students (5 females and 12 males) volunteered from the "Introduction to Information Security and Privacy" class. They have all taken the survey before this experiment.

Materials: Since this experiment requires subjects to study an entire protocol carefully, we have selected four short protocols: one contains 7 messages and the other three contains 5 messages each. Also, all of the messages in these protocols consist of a small number of primitives. Figure 70 shows these four protocols, corresponding



```
Question 1:
A->S:A,B
S->A:{Ku_B,B}Ks_AS
A->B:{N_A,A}Ku_B
B->S:B,A
S->B:{Ku_A,A}Ks_AS
B->A:{N_A,N_B}Ku_A
A->B:{N_B}Ku_B
Ans: a
a. A->B:{N_A,A}Ku_B
b. B->A:{N_B,A}Ku_B
c. B->S:{N_S,B}Ku_A
d. A->B:{N_B,B}Ku_A

Question 2:
A->S:A,B,N_A
S->A:{N_A,B,Ks_AB,{Ks_AB,A}Ks_BS}Ks_AS
A->B:{Ks_AB,A}Ks_BS
B->A:{N_B}Ks_AB
A->B:{N_B}Ks_AB
Ans: d
a. B->S:{Ks_BS,A}Ks_AB
b. A->B:{Ks_AS,S}Ks_AS
c. A->S:{Ks_AS,B}Ks_AS
d. A->B:{Ks_AB,A}Ks_BS

Question 3:
A->B:A
B->A:N_B
A->B:{N_B}Ks_AS
B->S:{A,{N_B}Ks_AS}Ks_BS
S->B:{N_B}Ks_BS
Ans: c
a. B->S:{B,{N_A}K_BS}K_AS
b. S->B:{A,{N_B}K_BS}K_AS
c. B->S:{A,{N_B}K_AS}K_BS
d. A->S:{B,{N_A}K_AS}K_BS

Question 4:
A->B:A
B->A:N_B
A->B:{N_B}Ks_AS
B->S:{A,{N_B}Ks_AS}Ks_BS
S->B:{A,N_B}Ks_BS
Ans: a
a. S->B:{A, N_B}Ks_BS
b. B->S:{A, N_B}Ks_AS
c. S->B:{B, N_A}Ks_AS
d. A->B:{B, N_A}Ks_BS
```

Figure 70: Four identification questions with choices and answers.

multiple choice questions, and their answers.

Procedure: To avoid the factor of question orders influencing the user study results, we adopt the following procedure. For each subject, our experiment environment first randomly divides the four protocols into two sets, one for the text-based approach and the other for the Lego-based approach, and randomly determines the sequences of protocols in each set. The order of the two approaches is also randomly chosen. Our experiment environment automatically uses the first approach on the first protocol set and the second approach on the second set.

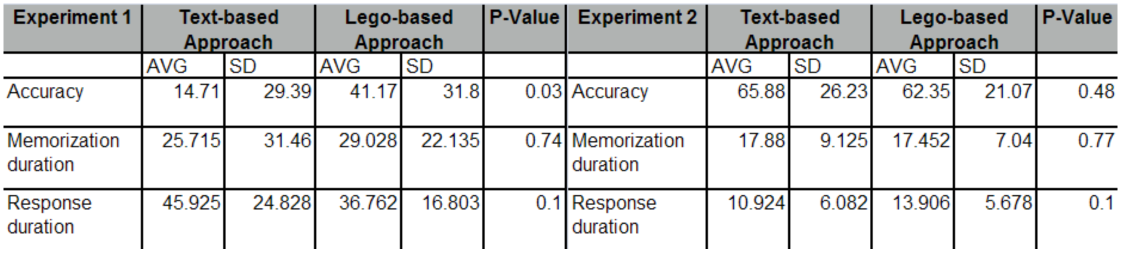
For each protocol, the system first enters a memorization phase, which allows a subject to study the protocol as long as he or she needs. Then, when the subjects indicate that they are ready, the system shows them one portion of the protocol (with three incorrect alternatives) in the same form that they have been viewing (digital or traditional) and asks them to identify the message that appears in the full protocol previously displayed. This leads to the response phase. After the subjects choose their answers and click the "next" button, our system displays the next question and repeats the same procedure until the experiment is finished. During the experiment, our system automatically records the accuracy, memorization duration, and response duration for each subject and each question.

Experiment Results and Analysis: Figures 71 (left) and 72 show the statistical results of this experiment. We calculate the averages and standard deviations of the accuracy, memorization duration, and response duration respectively.

From the results, we can see that the memorization and response durations for these two approaches are similar. The Lego-based approach attracts the attention just a little bit longer than the text-based approach (3.3 seconds). The p-value from t-test is 0.74 showing that this is not significantly different.

The response duration of the Lego-based approach is 9.2 seconds shorter than that of the text-based approach, indicating that the Lego-based approach may be easier for subjects to identify the missing primitives. However, the p-value from t-test is 0.1 showing that this difference is not significant.

The average accuracy of the Lego-based approach is much higher than that of the text-based approach. We think that the low accuracy of the text-based approach shows that the subjects have some difficulties on using the traditional method to



Experiment 1	Text-based Approach		Lego-based Approach		P-Value	Experiment 2	Text-based Approach		Lego-based Approach		P-Value
	AVG	SD	AVG	SD			AVG	SD	AVG	SD	
Accuracy	14.71	29.39	41.17	31.8	0.03	Accuracy	65.88	26.23	62.35	21.07	0.48
Memorization duration	25.715	31.46	29.028	22.135	0.74	Memorization duration	17.88	9.125	17.452	7.04	0.77
Response duration	45.925	24.828	36.762	16.803	0.1	Response duration	10.924	6.082	13.906	5.678	0.1

Figure 71: The results of experiment 1 (left) and experiment 2 (right).

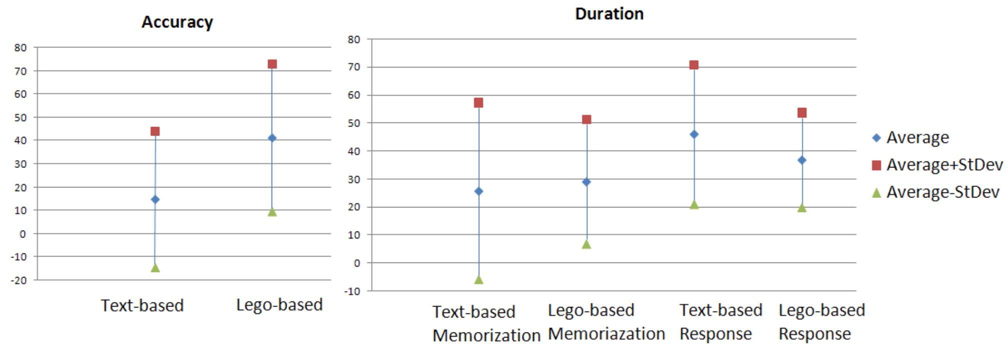
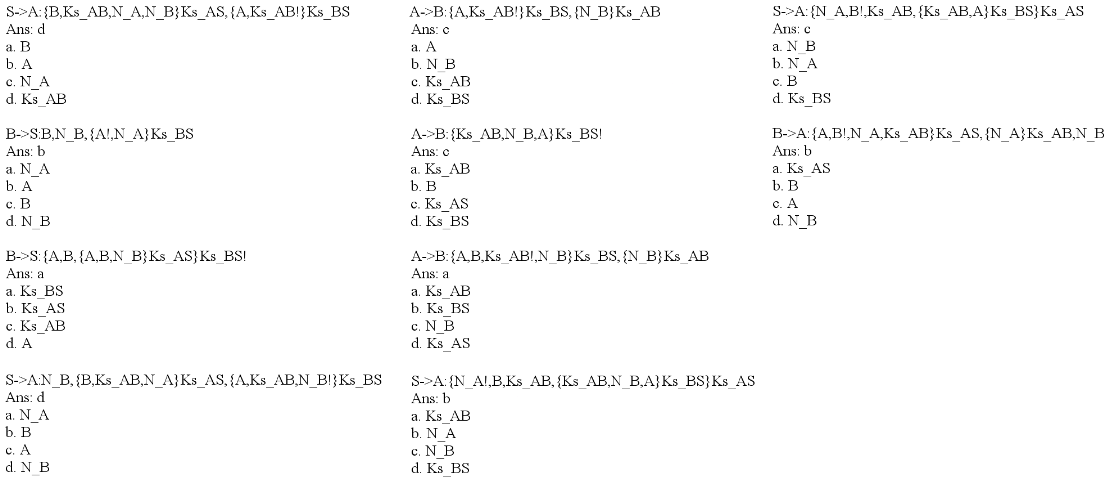


Figure 72: Data analysis results for experiment 1. The left shows the accuracy results and the right shows the memorization and response duration results.

identify the missing primitives. Since they have been familiar with the text-based approach throughout the semester, this may reflect some obstacles they have during the course. The absolute accuracy value of the Lego-based approach is also low, but its response duration is shorter and the accuracy is much higher than the text-based approach. The p-value from t-test is 0.03 showing that they are significantly different. We think that this result demonstrates one advantage of the Lego-based approach over the traditional text-based approach.

8.6.2.2 Experiment 2: Protocol Primitive Memorization

Since visual-based approaches might be used to strengthen user memory, we design this experiment to evaluate whether or not the Lego-based approach can help subjects remember the primitives in a protocol better than the text-based approach.



S->A:{B,Ks_AB,N_A,N_B}Ks_AS,{A,Ks_AB!}Ks_BS
Ans: d
a. B
b. A
c. N_A
d. Ks_AB

B->S:B,N_B,{A!,N_A}Ks_BS
Ans: b
a. N_A
b. A
c. B
d. N_B

B->S:{A,B,{A,B,N_B}Ks_AS}Ks_BS!
Ans: a
a. Ks_BS
b. Ks_AS
c. Ks_AB
d. A

S->A:N_B,{B,Ks_AB,N_A}Ks_AS,{A,Ks_AB,N_B!}Ks_BS
Ans: d
a. N_A
b. B
c. A
d. N_B

A->B:{A,Ks_AB!}Ks_BS,{N_B}Ks_AB
Ans: c
a. A
b. N_B
c. Ks_AB
d. Ks_BS

A->B:{Ks_AB,N_B,A}Ks_BS!
Ans: c
a. Ks_AB
b. B
c. Ks_AS
d. Ks_BS

A->B:{A,B,Ks_AB!,N_B}Ks_BS,{N_B}Ks_AB
Ans: a
a. Ks_AB
b. Ks_BS
c. N_B
d. Ks_AS

S->A:{N_A!,B,Ks_AB,{Ks_AB,N_B,A}Ks_BS}Ks_AS
Ans: b
a. Ks_AB
b. N_A
c. N_B
d. Ks_BS

S->A:{N_A,B!,Ks_AB,{Ks_AB,A}Ks_BS}Ks_AS
Ans: c
a. N_B
b. N_A
c. B
d. Ks_BS

B->A:{A,B!,N_A,Ks_AB}Ks_AS,{N_A}Ks_AB,N_B
Ans: b
a. Ks_AS
b. B
c. A
d. N_B

Figure 73: Ten memorization questions with choices and answers. The primitives followed by a "!" will be replaced by a "?" in the experiments.

Apparatus and Subjects: The same as Experiment 1.

Materials: We have selected ten messages with different kinds of primitives appearing in general security protocols. They are neither too long nor too short. The average number of primitives in these messages is 6. Figure 73 shows the ten messages, corresponding multiple choice questions, and their answers.

Procedure: The same as experiment 1, our experiment environment randomly divides all the messages into two sets, one for the text-based approach and the other for the Lego-based approach, and randomly determines the question sequences in each set. The order of the two approaches is also randomly determined.

The experiment procedure for each message is similar to the procedure for each protocol in the experiment 1. After subjects study a message and click the question button, our experiment environment replaces one primitive in the message with a "?" mark, and displays four choices in the same format as the message. Figure 65 shows one example of a message in the experiment. The subjects are then asked to identify which primitive has appeared in the previous message. This procedure is repeated until all the questions have been answered. During the experiment, we record the accuracy, memorization duration and response duration for each subject and each question.

Experiment Results and Analysis: Figures 71 (right) and 74 show the statistical results of this experiment. We calculate the averages and standard deviations of the accuracy, memorization duration, and response duration respectively.

All the results, including the accuracy, memorization duration, and response duration, of these two approaches are similar. The p-values from the t-test also show that they are not significantly different. The small difference between these two approaches may come from the fact that the text-based approach has been used to teach these subjects throughout the semester, while the Lego-based approach is only briefly introduced before the experiment.

Combining the results from our survey and two user studies, we think that the Lego-based approach obviously offers more meaningful and interesting information for students to observe, especially on the relationships among primitives and protocols. Suitable usages of such visual information may lead to direct benefits for students to learn and apply security protocols.

8.7 Discussion

The main strengths of the Lego-based approach are two fold: attract the interests of students and improve the understanding of security protocols. First, it is essential to the success of information assurance education that we can attract and retain the

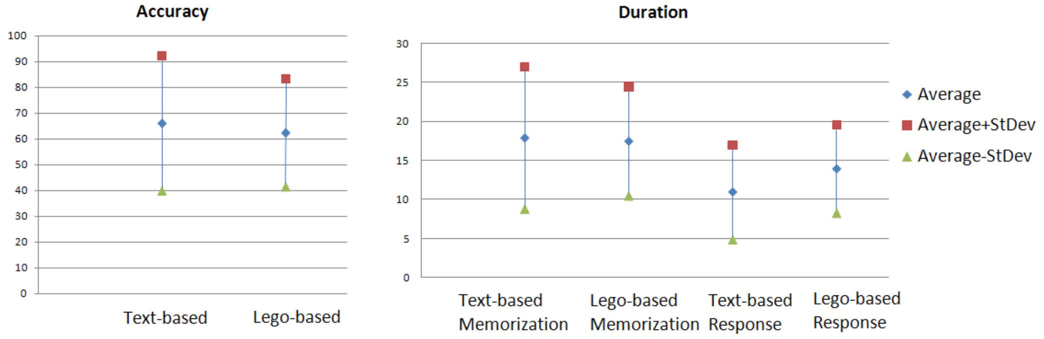


Figure 74: Data analysis results for experiment 2. The left shows the accuracy results and the right shows the memorization and response duration results.

interests of students. Both the survey and user studies in our evaluation demonstrate that the combination of 3D digital Legos and the text-based approach is the best solution for students to accept. We also emphasize this strength by providing several features to our Lego system, including the flexibility to change primitive designs and the 3D interaction methods that simulate real-life Lego experiences.

Second, the Lego toys promote children to recognize individual shapes and the matching relationships among different blocks. Similarly, our approach constructs digital Legos to help students identify individual security primitives and improve their understanding of the relationships among primitives and protocols. Our user studies evaluate two important aspects, primitive identification and memorization, since they are directly related to our objectives. A good understanding of the relationships among security primitives and protocols cannot be separated from the understanding of individual primitives. During our development process, we explore different designs of primitives, such as the shapes of key words and similar shapes from objects in real-life, to help students link the protocol contents to the shapes of the Lego blocks. The evaluation results demonstrate significantly better primitive identification performance of our Lego-based approach compared to the traditional text-based approach. We believe that once students are familiar with the primitive pieces, better recognition can lead to better memorization of protocol details, and thereby improving the understanding of security protocols. We plan to design more user studies to evaluate other aspects of protocol understanding in the future.

In addition to these impacts, our approach also has the potential to help students understand the linkage between the protocol design and its vulnerabilities. Here we use the man-in-the-middle attack as an example to illustrate the potential. A security protocol is vulnerable to the man-in-the-middle attack when the receiver cannot verify the authenticity and integrity of a message. For example, when A sends its identity and public key in plain-text to B, an attacker on the path can switch A's public key with its own public key. Under this attack, any messages that B intends to send to A can be read by the attacker. We have integrated our digital Lego system with the knowledge model for security protocols [91] to illustrate these attacks. As shown in Figure 62, for every entity both its initial knowledge when the protocol starts and the latest knowledge as the protocol proceeds are shown on the left bottom panel. Therefore, we can combine the content of a message and the latest knowledge of its receiver to identify the components that the receiver cannot verify or authenticate. These components are then labeled in a special color to show that an attacker could have changed their values and a man-in-the-middle attack might exist. Note that this functionality is not dependent on any specific protocols. In fact, we have adopted this technique in our undergraduate level security course to allow the students to understand and compare the man-in-the-middle attacks and type flaw attacks on the key exchange protocols such as Diffie-Hellman and Needham-Schroeder public key protocols.

8.8 Conclusion and Future Work

To improve the information assurance education, we have developed a digital Lego system for demonstrating and practicing important security concepts. We carefully design our digital Lego sets to provide a generic representation of security protocols. Our approach applies the pedagogical methods learned from toy construction sets by treating security primitives as Lego pieces and protocols as construction results. With our digital Lego sets, we have developed a prototype system and supporting instructional materials. We have also designed and performed evaluations to assess this Lego-based approach and found encouraging results and feedback.

In the future, we plan to introduce our digital Lego approach and course materials gradually into the introductory level security courses. We have collected a list of security protocols that are widely adopted in information assurance education. We will apply interactive visualization techniques to develop supporting functions and integrate them into a more comprehensive experiment environment. We plan to publish our course materials and Lego system online to share with other researchers and educators. We will also continue to perform formal user studies to gather data from larger groups and evaluate the effectiveness of the Lego-based approach on aiding students to understand security protocols. The results of the user studies will be used to improve our Lego-based approach, so that security knowledge can be introduced to a broader population.

CHAPTER 9: CONCLUSIONS AND FUTURE WORK

9.1 Conclusions

This dissertation presents new abstract visualization approaches for visualizing large-scale time-varying datasets. Our concentration on abstract approaches is rooted from the characteristics of temporal relationships. Two important components, data feature or event extraction and statistical sampling, provide the essential techniques to measure temporal relationships. They are also crucial to handle large-scale data visualization. We believe that abstract visualization suits for the requirements of applications of large-scale time-varying datasets.

Specifically, we present two abstract visualization platforms: time line and animated visualization. The time line approach transforms temporal relationships to simple lines, which are generally easy to understand ans study. The animated visualization approach summarizes a time-varying dataset as an event tree, which also visualizes the composition structure of the focused event. Generally speaking, time line provides an abstract snapshot of the entire time-varying dataset; while animated visualization presents the time-varying dataset from the view of features-of-interest. The two visualization platforms can be combined to visualize details at different levels.

9.2 Future Work

Large-scale time-varying data visualization is a very challenging topic. While efforts including this dissertation have been made to develop effective approaches, there are still gaps between current visualization techniques and expectations of scientific users. Specifically related to the work presented in this dissertation, our future work can be summarized under the following topics:

The approach of time lines can be used for other types of datasets, such as vector time-varying datasets and videos. We are interested in investigating solutions to improve the effectiveness of time-varying data visualization from the following aspects. First, we believe that the concept of virtual words can be used to improve the understanding of time-varying datasets in more flexible ways. We plan to design a multi-link interface that can incorporate more information from virtual words to provide new visual analytics capabilities. Second, we are interested in embedding additional advanced comparison and analysis tools that use time lines as an interface to visualize time-varying datasets. Third, we plan to integrate some useful information we collect from feature points and feature descriptors in our time line visualization. We are also interested in improving the low-dimensional embedding algorithms for time line generation.

There are several future direction for our ensemble visualization. First, we plan to build interactive and multi-display ensemble system to allow scientists explore and analyze ensemble dataset through more information visualization technologies to find more insights. Second, we intend to apply our methods to volumetric simulations.

This may require more pre-processing of the dataset and more sophisticated feature extraction and description methods. Third, more information could be included in the final visualization. Since time line is mainly an abstract shape to show how data changes, given more 3D geo-spatial renderings can assist scientists to better explore their dataset. Lastly, uncertainty, input/output parameter effects and more summary statistics analysis need to be included into the study process to help scientist understand not only the outcomes of simulation but also the underlying distribution of members.

The approach of animated visualization has received positive feedbacks. Our future work includes exploring the effectiveness of different narrative structures of the same event given summary or interactive exploration tasks. This result can provide useful information to design general animation techniques. We are also interested in extending this approach to volumetric time-varying data visualization, where we expect to integrate more feature tracking and data reduction methods.

For the evaluation of animated visualization, we plan to continue the study with students and faculties from Meteorology Department to learn if there are differences in accuracy from people who had more knowledge on storm surges. Secondly, we want to test more features and phenomena that scientists are interested in as well as design more study categories for visual analysis purposes. We plan to bring multiple simulations together and improve the animation generation approach to fit these requirements. Lastly, we will try different scientific datasets besides storm surge simulations to see how users perceive knowledge from other datasets.

REFERENCES

[1] http://www.emc.ncep.noaa.gov/index.php?branch=SREF.

[2] http://www.cmascenter.org/.

[3] http://www.smoke-model.org/index.cfm.

[4] http://www.adcirc.org/.

[5] http://www.unidata.ucar.edu/software/netcdf/.

[6] Aigner, W., Miksch, S., Müller, W., Schumann, H., and Tominski, C. Visual methods for analyzing time-oriented data. IEEE Transactions on Visualization and Computer Graphics 14, 1 (2008), 47–60.

[7] Aigner, W., Rind, A., and Hoffmann, S. Comparative Evaluation of an Interactive Time-Series Visualization that Combines Quantitative Data with Qualitative Abstractions. Computer Graphics Forum 31, 3 (2012), 995–1004.

[8] Ainsworth, S., and VanLabeke, N. Multiple forms of dynamic representation. Learning and Instruction 14, 3 (June 2004), 241–255.

[9] Akiba, H., Fout, N., and Ma, K.-L. Simultaneous classification of time-varying volume data based on the time histogram. In EuroVis (2006), pp. 171–178.

[10] Akiba, H., and Ma, K.-L. A tri-space visualization interface for analyzing time-varying multivariate volume data. In Proceedings of The Joint Eurographics-IEEE VGTC Symposium on Visualization (2007).

[11] Akiba, H., Wang, C., and Ma, K.-L. Aniviz: A template-based animation tool for volume visualization. IEEE Computer Graphics and Applications 99 (2009).

[12] Anselin, L., Syabri, I., and Smirnov, O. Visualizing multivariate spatial correlation with dynamically linked windows. In University of California, Santa Barbara. CD-ROM (2002).

[13] Archambault, D., Purchase, H., and Pinaud, B. Animation, small multiples, and the effect of mental map preservation in dynamic graphs. IEEE Transactions on Visualization and Computer Graphics 17, 4 (Apr. 2011), 539–552.

[14] Assa, J., Caspi, Y., and Cohen-Or, D. Action synopsis: Pose selection and illustration. In Proceedings of ACM SIGGRAPH (2005), pp. 667–676.

[15] Bachthaler, S., and Weiskopf, D. Animation of orthogonal texture patterns for vector field visualization. IEEE Transactions on Visualization and Computer Graphics 14, 4 (2008), 741–755.

[16] Balabanian, J.-P., Viola, I., Möller, T., and Gröller, E. Temporal styles for time-varying volume data. In Proceedings of 3DPVT'08 - the Fourth International Symposium on 3D Data Processing, Visualization and Transmission (June 2008), S. Gumhold, J. Kosecka, and O. Staadt, Eds., pp. 81–89.

[17] Banks, D. C., and Singer, B. A. A predictor-corrector technique for visualizing unsteady flow. IEEE Transactions on Visualization and Computer Graphics 1, 2 (1995), 151–163.

[18] Bay, H., Ess, A., Tuytelaars, T., and Van Gool, L. Speeded-up robust features (surf). Comput. Vis. Image Underst. 110 (June 2008), 346–359.

[19] Berlin, B., and Kay, P. Basic Color Terms: Their Universality and Evolution. The David Hume Series, 1991.

[20] Blaas, J., Botha, C., and Post, F. Extensions of parallel coordinates for interactive exploration of large multi-timepoint data sets. Visualization and Computer Graphics, IEEE Transactions on 14, 6 (nov.-dec. 2008), 1436 –1451.

[21] Blok, C. A. Interactive animation to visually explore time series of satellite imagery. In Proceedings of the 8th international conference on Visual Information and Information Systems (Berlin, Heidelberg, 2006), VISUAL'05, Springer-Verlag, pp. 71–82.

[22] Blumenkrants, M., Starovisky, H., and Shamir, A. Narrative algorithm visualization. In SoftVis '06: Proceedings of the 2006 ACM symposium on Software visualization (2006).

[23] Bly, B. M., and Rumelhart, D. E. Cognitive Science (Handbook of Perception and Cognition). Academic Press, 1999.

[24] Borg, I., and Groenen, P. Modern Multidimensional Scaling: Theory and Applications. Springer, 1997.

[25] Boyandin, I., Bertini, E., and Lalanne, D. A Qualitative Study on the Exploration of Temporal Changes in Flow Maps with Animation and Small-Multiples. Computer Graphics Forum 31, 3 (2012), 1005–1014.

[26] Brown, M., and Lowe, D. G. Recognising panoramas. In ICCV '03: Proceedings of the Ninth IEEE International Conference on Computer Vision (2003), p. 1218.

[27] Bruckner, S., and Möandller, T. Result-driven exploration of simulation parameter spaces for visual effects design. Visualization and Computer Graphics, IEEE Transactions on 16, 6 (nov.-dec. 2010), 1468 –1476.

[28] Burger, C., and Rothermel, K. A framework to support teaching in distributed systems. Journal on Educational Resources in Computing 1, 3 (2001).

[29] Caban, J. J., and Rheingans, P. Texture-based transfer functions for direct volume rendering. IEEE Transactions on Visualization and Computer Graphics 14, 6 (2008), 1364–1371.

[30] Chen, C.-K., Wang, C., Ma, K.-L., and Wittenberg, A. Static correlation visualization for large time-varying volume data. In Proceedings of IEEE Pacific Visualization Symposium (March 2011), pp. 27–34.

[31] Chiueh, T.-c., and Ma, K.-L. A parallel pipelined renderer for time-varying volume data. In ISPAN '97: Proceedings of the 1997 International Symposium on Parallel Architectures, Algorithms and Networks (Washington, DC, USA, 1997), IEEE Computer Society, p. 9.

[32] Chuang, J., Weiskopf, D., and Moller, T. Hue-preserving color blending. IEEE Transactions on Visualization and Computer Graphics 15, 6 (Nov. 2009), 1275–1282.

[33] Cignoni, P., and et al. Meshlab. http://meshlab.sourceforge.net/.

[34] Comaniciu, D., and Meer, P. Mean shift: A robust approach toward feature space analysis. IEEE Transactions on Pattern Analysis and Machine Intelligence 24, 5 (2002), 603–619.

[35] Cooten, V., Suzanne, and Coauthors. The ci-flow project: A system for total water level prediction from the summit to the sea. Bulletin of the American Meteorological Society 92 (2011), 1427šC1442.

[36] Cormen, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C. Introduction to Algorithms. The MIT Press, 2001.

[37] Correa, C. D., and Silver, D. Dataset traversal with motion-controlled transfer functions. Visualization Conference, IEEE 0 (2005), 46.

[38] Coulston, C., and Ford, R. Teaching functional decomposition for the design of electrical and computer systems. In Proc. of IEEE Frontiers in Education Annual Conference (2004).

[39] Cremers, C. Compositionality of security protocols: A research agenda. Electronic Notes in Theoretical Computer Science 142, 3 (2006), 99–110.

[40] Cross, G. Kids' Stuff. Harvard University Press, 1997.

[41] Dietrich, J., Bunya, S., Westerink, J., Ebersole, B., Smith, J., Atkinson, J., RE Jensen, D. R., Jr, R. L., Dawson, C., Cardone, V., Cox, A., Powell, M., Westerink, H., and Roberts, H. A high-resolution coupled riverine flow, tide, wind, wind wave, and storm surge model for southern louisiana and mississippi, part ii: Synoptic description and analysis of hurricanes katrina and rita. Monthly Weather Review 138 (2010), 378–404.

[42] Dietrich, J., Tanaka, S., Westerink, J., Dawson, C., Jr, R. L., Zijlema, M., Holthuijsen, L., Smith, J., Westerink, L., and Westerink, H. Performance of the unstructured-mesh, swan+adcirc model in computing hurricane waves and surge. Journal of Scientific Computing 52(2) (2012), 468–497.

[43] Dietrich, J., Trahan, C., Howard, M., Fleming, J., Weaver, R., Tanaka, S., Yu, L., Jr., R. L., Dawson, C., Westerink, J., Wells, G., Lu, A., Vega, K., Kubach, A., Dresback, K., Kolar, R., Kaiser, C., and Twilley, R. Surface trajectories of oil transport along the northern coastline of the gulf of mexico. Continental Shelf Research 41, 0 (2012), 17 – 47.

[44] Dietrich, J., Zijlema, M., Westerink, J., Holthuijsen, L., Dawson, C., Jr, R. L., Jensen, R., Smith, J., Stelling, G., and Stone, G. Modeling hurricane waves and storm surge using integrally-coupled, scalable computations. Coastal Engineering 58 (2011), 45–65.

[45] Dolev, D., and Yao, A. On the security of public key protocols. IEEE Transactions on Information Theory 29, 2 (1983), 198–208.

[46] Du, Z., Chiang, Y.-J., and Shen, H.-W. Out-of-core volume rendering for time-varying fields using a space-partitioning time (spt) tree. In Visualization Symposium, 2009. PacificVis '09. IEEE Pacific (april 2009), pp. 73 –80.

[47] Edelsbrunner, H., Harer, J., Mascarenhas, A., and Pascucci, V. Time-varying reeb graphs for continuous space-time data. In Proceedings of 20th Ann. Sympos. Comput. Geom. (2004), pp. 366–372.

[48] Eisenberg, M., Eisenberg, A., Gross, M., Kaowthumrong, K., Lee, N., and Lovett, W. Computationally-enhanced construction kits for children: Prototype and principles. In Proc. of Int. Conf. of Learning Sciences (2002), pp. 79–85.

[49] Elmqvist, N. Protoviz: A simple security protocol visualization. Tech. rep., the University of Gothenburg, 2004.

[50] Elmqvist, N., Dragicevic, P., and Fekete, J.-D. Rolling the dice: Multidimensional visual exploration using scatterplot matrix navigation. IEEE Trans. Vis. Comput. Graph. 14, 6 (2008), 1539–1148.

[51] Fang, Z., Möller, T., Hamarneh, G., and Celler, A. Visualization and exploration of time-varying medical image data sets. In Proceedings of Graphics Interface 2007 (New York, NY, USA, 2007), GI '07, ACM, pp. 281–288.

[52] Farrugia, M., and Quigley, A. Effective temporal graph layout: a comparative study of animation versus static display methods. Journal of Information Visualization (2011).

[53] Finkelstein, A., Jacobs, C. E., and Salesin, D. H. Multiresolution video. In SIGGRAPH '96: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques (1996).

[54] Fischer, F., Fuchs, J., and Mansmann, F. ClockMap: Enhancing Circular Treemaps with Temporal Glyphs for Time-Series Data. In Proceedings of the Eurographics Conference on Visualization (EuroVis 2012 Short Papers) (Vienna, Austria, 2012), M. Meyer and T. Weinkauf, Eds., pp. 97–101.

[55] Fisher, R. The Statistical Utilization of Multiple Measurements. Annals of Eugenics 8 (1938), 376–386.

[56] Fuchs, R., and Hauser, H. Visualization of multi-variate scientific data. Computer Graphics Forum 28, 6 (2009), 1670–1690.

[57] Fujishiro, I., Otsuka, R., Takeshima, Y., and Takahashi, S. T-map: A topological approach to visual exploration of time-varying volume data. In Proceedings of ISHPC2005, Springer Lecture Notes in Computer Science (2007), vol. 4759.

[58] Fukunaga, K. Introduction to statistical pattern recognition (2nd ed.). Academic Press Professional, Inc., San Diego, CA, USA, 1990.

[59] Georgescu, B., Shimshoni, I., and Meer, P. Mean shift based clustering in high dimensions: A texture classification example. In International Conference on Computer Vision (2003), pp. 456–463.

[60] Gershon, N., and Page, W. What storytelling can do for information visualization. Commun. ACM 44, 8 (2001), 31–37.

[61] Gershon, N. D. Visualization of fuzzy data using generalized animation. In VIS '92: Proceedings of the 3rd conference on Visualization '92 (1992), pp. 268–273.

[62] Gerstner, T., and Pajarola, R. Topology preserving and controlled topology simplifying multiresolution isosurface extraction. In Proceedings of Visualization (2000), pp. 259–266.

[63] Glatter, M., Huang, J., Ahern, S., Daniel, J., and Lu, A. Visualizing temporal patterns in large multivariate data using modified globbing. Visualization and Computer Graphics, IEEE Transactions on 14, 6 (nov.-dec. 2008), 1467 –1474.

[64] Glatter, M., Huang, J., Ahern, S., Daniel, J., and Lu, A. Visualizing temporal patterns in large multivariate data using textual pattern matching. IEEE Transactions on Visualization and Computer Graphics 14, 6 (2008), 1467–1474.

[65] Gneiting, T., and Raftery, A. E. Weather forecasting with ensemble methods. Science 310 (October 2005), 248–24.

[66] Goldfeather, J., and Interrante, V. A novel cubic-order algorithm for approximating principal direction vectors. ACM Trans. Graph. 23, 1 (2004), 45–63.

[67] Hao, M. C., Janetzko, H., Mittelstädt, S., Hill, W., Dayal, U., Keim, D. A., Marwah, M., and Sharma, R. K. A visual analytics approach for peak-preserving prediction of large seasonal time series. Comput. Graph. Forum (2011), 691–700.

[68] Harley, B. Constructional Toys. Shire Publications, UK, 1990.

[69] Harris, C., and Stephens, M. A combined corner and edge detector. In Proceedings of the 4th Alvey Vision Conference (1988), pp. 147–151.

[70] Heer, J., and Robertson, G. Animated transitions in statistical data graphics. IEEE Transactions on Visualization and Computer Graphics 13, 6 (Nov. 2007), 1240–1247.

[71] Hegarty, M. Dynamic visualizations and learning: getting to the difficult questions. Learning and Instruction 14, 3 (June 2004), 343–351.

[72] Howe, A. The ultimate construction toy: Applying kit-ofparts theory to habitat and vehicle design. In Proc. of Aerospace Architecture Symposium (2002).

[73] Hsu, W.-H., Mei, J., Correa, C., and Ma, K.-L. Depicting time evolving flow with illustrative visualization techniques. 1st International ICST Conference on Arts and Technology (May 2009).

[74] Jacquemard, F., and et al. Security protocols open repository. http://www.lsv.ens-cachan.fr/Software/spore/index.html.

[75] Jang, Y., Ebert, D. S., and Gaither, K. Time-varying data visualization using functional representations. IEEE Transactions on Visualization and Computer Graphics 18 (2012), 421–433.

[76] Janicke, H., Bottinger, M., Mikolajewicz, U., and Scheuermann, G. Visual exploration of climate variability changes using wavelet analysis. Visualization and Computer Graphics, IEEE Transactions on 15, 6 (nov.-dec. 2009), 1375 –1382.

[77] Jankun-Kelly, T. J., and Ma, K.-L. A study of transfer function generation for time-varying volume data. In Proceedings of Volume Graphics (2001), pp. 51–65.

[78] Ji, G., Shen, H.-W., and Wenger, R. Volume tracking using higher dimensional isosurfacing. In Proceedings of IEEE Visualization (2003).

[79] Johnson, C., and Hansen, C. Visualization Handbook. Academic Press, Inc., Orlando, FL, USA, 2004.

[80] Jolliffe, I. T. Principal Component Analysis. Springer, New York, NY, USA, 2002.

[81] Joshi, A., Caban, J., Rheingans, P., and Sparling, L. Case study on visualizing hurricanes using illustration-inspired techniques. IEEE Transactions on Visualization and Computer Graphics 15, 5 (2009), 709–718.

[82] Joshi, A., and Rheingans, P. Evaluation of illustration-inspired techniques for time-varying data visualization. Comput. Graph. Forum 27, 3 (2008), 999–1006.

[83] Keefe, D., Ewert, M., Ribarsky, W., and Chang, R. Interactive coordinated multiple-view visualization of biomechanical motion data. Visualization and Computer Graphics, IEEE Transactions on 15, 6 (nov.-dec. 2009), 1383 –1390.

[84] Kehoe, C., Stasko, J., and Taylor, A. Rethinking the evaluation of algorithm animations as learning aids: an observational study. Int. J. Hum.-Comput. Stud. 54, 2 (Feb. 2001), 265–284.

[85] Kehrer, J., Muigg, P., Doleisch, H., and Hauser, H. Interactive visual analysis of heterogeneous scientific data across an interface. Visualization and Computer Graphics, IEEE Transactions on 17, 7 (july 2011), 934 –946.

[86] Kindlmann, G., and Durkin, J. W. Semi-automatic generation of transfer functions for direct volume rendering. In IEEE Symposium on Volume Visualization (1998), pp. 79–86.

[87] Kitamura, Y., and et al. Real-time 3d interaction with activecube. In Proc. of CHI (2001), pp. 355–356.

[88] Lazebnik, S., Schmid, C., and Ponce, J. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on (2006), vol. 2, pp. 2169–2178.

[89] Lee, T.-Y., and Shen, H.-W. Visualization and exploration of temporal trend relationships in multivariate time-varying data. IEEE Transactions on Visualization and Computer Graphics 15, 6 (2009), 1359–1366.

[90] Lefer, W., Jobard, B., and Leduc, C. High-quality animation of 2d steady vector fields. IEEE Transactions on Visualization and Computer Graphics 10, 1 (2004), 2–14.

[91] Li, Z., and Wang, W. Using deductive knowledge to improve cryptographic protocol verification. In IEEE Military Communications Conference (MILCOM) (2009).

[92] Lowe, D. G. Object recognition from local scale-invariant features. Computer Vision, IEEE International Conference on 2 (1999), 1150.

[93] Lu, A., Maciejewski, R., and Ebert, D. S. Volume composition using eye tracking data. In Proceedings of EuroVis (2006).

[94] Lu, A., and Shen, H.-W. Interactive storyboard for overall time-varying data visualization. In Proceedings of IEEE Pacific Visualization Symposium (2008), pp. 143–150.

[95] Lum, E. B., Stompel, A., and Ma, K. L. Kinetic visualization: a technique for illustrating 3d shape and structure. In VIS '02: Proceedings of the conference on Visualization '02 (2002), pp. 435–442.

[96] Lundström, C., Ljung, P., Persson, A., and Ynnerman, A. Uncertainty visualization in medical volume rendering using probabilistic animation. IEEE Transactions on Visualization and Computer Graphics 13, 6 (2007), 1648–1655.

[97] Maloney, J., Burd, L., Kafai, Y., Rusk, N., Silverman, B., and Resnick, M. Scratch: A sneak preview. In Int. Conf. on Creating, Connecting, and Collaborating through Computing (2004), pp. 104–109.

[98] Mao, Y., Dillon, J., and Lebanon, G. Sequential document visualization. IEEE Transactions on Visualization and Computer Graphics 13, 6 (2007), 1208–1215.

[99] Martin, F., Mikhak, B., Resnick, M., Silverman, B., and Berg, R. To mindstorms and beyond: evolution of a construction kit for magical machines. In Robots for kids: exploring new technologies for learning (2000), pp. 9–33.

[100] Millen, J., and Shmatikov, V. Constraint solving for bounded-process cryptographic protocol analysis. In Proc. of ACM CCS (2001), pp. 166–175.

[101] Nam, J. E., Maurer, M., and Mueller, K. Knowledge assisted visualization: A high-dimensional feature clustering approach to support knowledge-assisted visualization. Comput. Graph. 33, 5 (2009), 607–615.

[102] Ni, D., Chui, Y. P., Qu, Y., Yang, X., Qin, J., Wong, T.-T., Ho, S. S. H., and Heng, P. A. Reconstruction of volumetric ultrasound panorama based on improved 3d sift. Computerized Medical Imaging and Graphics (June 2009).

[103] Obstfeld, R. Fiction First Aid: Instant Remedies for Novels, Stories and Scripts. Writer's Digest Books, Cincinnati, OH, 2002.

[104] Phadke, M. N., Pinto, L., Alabi, F., Harter, J., 2nd, R. M. T., Wu, X., Hannah Petersen, S. A. B., and Healey, C. G. Exploring ensemble visualization. In Visualization and Data Analytics 2012 (2012), pp. 1–12.

[105] Post, F. H., Vrolijk, B., Hauser, H., Laramee, R. S., and Doleisch, H. The state of the art in flow visualization: Feature extraction and tracking. Computer Graphics Forum 22, 4 (2003), 775–792.

[106] Pothamsetty, V. Where security education is lacking. In Proc. of InfoSecCD (2005), pp. 54–58.

[107] Potter, K., Wilson, A., Bremer, P.-T., Williams, D., Doutriaux, C., Pascucci, V., and Johhson, C. R. Visualization of uncertainty and ensemble data: Exploration of climate modeling and weather forecast data with integrated visus-cdat systems. In Proceedings of SciDAC 2009 (2009), vol. 180 of Journal of Physics: Conference Series, p. (published online).

[108] Potter, K., Wilson, A., Bremer, P.-T., Williams, D., Doutriaux, C., Pascucci, V., and Johnson, C. Ensemble-vis: A framework for the statistical visualization of ensemble data. In Data Mining Workshops, 2009. ICDMW '09. IEEE International Conference on (dec. 2009), pp. 233 –240.

[109] Reinders, F., Post, F. H., and Spoelder, H. J. Visualization of time-dependent data using feature tracking and event detection. The Visual Computer 17, 1 (2001), 55–71.

[110] Resnick, M., and et al. Programmable bricks: Toys to think with. IBM Systems Journal 35, 3 (1996), 443–452.

[111] Robertson, G., Fernandez, R., Fisher, D., Lee, B., and Stasko, J. Effectiveness of animation in trend visualization. IEEE Transactions on Visualization and Computer Graphics 14, 6 (2008), 1325–1332.

[112] Robertson, G. G., Card, S. K., and Mackinlay, J. D. Information visualization using 3d interactive animation. Commun. ACM 36, 4 (1993), 57–71.

[113] Roweis, S., and Saul, L. Nonlinear dimensionality reduction by locally linear embedding. Science 290, 5500 (2000), 2323–2326.

[114] Samtaney, R., Silver, D., Zabusky, N., and Cao, J. Visualizing features and tracking their evolution. IEEE Trans. Comput. 27 (1994), 20–27.

[115] Sanyal, J., Zhang, S., Dyer, J., Mercer, A., Amburn, P., and Moorhead, R. Noodles: A tool for visualization of numerical weather model ensemble uncertainty. Visualization and Computer Graphics, IEEE Transactions on 16, 6 (nov.-dec. 2010), 1421 –1430.

[116] Saul, E., and Hutchison, A. A graphical environment for the facilitation of logic-based security protocol analysis. South African Computer, 21 (1998), 26–30.

[117] Schweitzer, D., Baird, L., Collins, M., Brown, W., and Sherman, M. Grasp: A visualization tool for teaching security protocols. In Proceedings of the 10th Colloquium for Information Systems Security Education (2006).

[118] Scovanner, P., Ali, S., and Shah, M. A 3-dimensional sift descriptor and its application to action recognition. In MULTIMEDIA '07: Proceedings of the 15th international conference on Multimedia (2007), pp. 357–360.

[119] Shen, H.-W. Isosurface extraction in time-varying fields using a temporal hierarchical index tree. In VIS '98: Proceedings of the conference on Visualization '98 (Los Alamitos, CA, USA, 1998), IEEE Computer Society Press, pp. 159–166.

[120] Shen, H.-W., Chiang, L.-J., and Ma, K.-L. A fast volume rendering algorithm for time-varying fields using a time-space partitioning (tsp) tree. In VISUALIZATION '99: Proceedings of the 10th IEEE Visualization 1999 Conference (VIS '99) (Washington, DC, USA, 1999), IEEE Computer Society.

[121] Shen, H.-W., Chiang, L.-J., and Ma, K.-L. A fast volume rendering algorithm for time-varying fields using a time-space partitioning (tsp) tree. In IEEE Visualization (1999).

[122] Shi, K., Theisel, H., Hauser, H., Weinkauf, T., Matkovic, K., Hege, H.-C., and Seidel, H.-P. Path line attributes – an information visualization approach to analyzing the dynamic behavior of 3d time-dependent flow fields. In Topology-Based Methods in Visualization II (2009), pp. 75–88.

[123] Silver, D., and Wang, X. Tracking and visualizing turbulent 3d features. IEEE Transaction on Visualization and Computer Graphics 3, 2 (1997), 129–141.

[124] Sohn, B.-S., and Bajaj, C. Time-varying contour topology. IEEE Transactions on Visualization and Computer Graphics 12, 1 (2006), 14–125.

[125] Stasko, J., Badre, A., and Lewis, C. Do algorithm animations assist learning?: an empirical study and analysis. In Proceedings of the INTERACT '93 and CHI '93 conference on Human factors in computing systems (New York, NY, USA, 1993), CHI '93, ACM, pp. 61–66.

[126] Sukharev, J., Wang, C., Ma, K.-L., and Wittenberg, A. T. Correlation study of time-varying multivariate climate data sets. In PACIFICVIS '09: Proceedings of the 2009 IEEE Pacific Visualization Symposium (2009), pp. 161–168.

[127] Talbot, J., Lee, B., Kapoor, A., and Tan, D. Ensemblematrix: Interactive visualization to support machine learning with multiple classifiers. In ACM Human Factors in Computing Systems (CHI) (2009).

[128] Thomas, J. J., and Cook, K. A., Eds. Illuminating the Path: The Research and Development Agenda for Visual Analytics. National Visualization and Analytics Ctr, 2005.

[129] Tikhonova, A., Correa, C., and Ma, K.-L. An exploratory technique for coherent visualization of time-varying volume data. Computer Graphics Forum 29, 3 (June 2010). (also Proceedings of EuroVis 2010).

[130] Torgeson, W. Multidimensional scaling of similarity. Psychometrika 30 (1965), 379–393.

[131] Trucco, E., and Verri, A. Introductory Techniques for 3-D Computer Vision. Prentice Hall PTR, Upper Saddle River, NJ, USA, 1998.

[132] Turkay, C., Parulek, J., Reuter, N., and Hauser, H. Interactive visual analysis of temporal cluster structures. Computer Graphics Forum 30, 3 (2011), 711–720.

[133] TVERSKY, B., MORRISON, J. B., and BETRANCOURT, M. Animation: can it facilitate? International Journal of Human-Computer Studies 57, 4 (2002), 247 – 262.

[134] Verma, V., and Pang, A. Comparative flow visualization. IEEE Transactions on Visualization and Computer Graphics 10, 6 (2004), 609–624.

[135] Viola, I., Feixas, M., Sbert, M., and Groller, M. E. Importance-driven focus of attention. IEEE Transactions on Visualization and Computer Graphics 12, 5 (2006), 933–940.

[136] Walker, J., Geng, Z., Jones, M., and Laramee, R. S. Visualization of Large, Time-Dependent, Abstract Data with Integrated Spherical and Parallel Coordinates. M. Meyer and T. Weinkauf, Eds., Eurographics Association, pp. 43–47.

[137] Wan, L., Mak, S.-K., Wong, T.-T., and Leung, C.-S. Spatiotemporal sampling of dynamic environment sequences. Visualization and Computer Graphics, IEEE Transactions on 17, 10 (oct. 2011), 1499 –1509.

[138] Wandell, B. A. Foundations of Vision. Sinauer Associates, 1995.

[139] Wang, C., and Chiang, Y.-J. Isosurface extraction and view-dependent filtering from time-varying fields using persistent time-octree (ptot). Visualization and Computer Graphics, IEEE Transactions on 15, 6 (nov.-dec. 2009), 1367 –1374.

[140] Wang, C., Yu, H., Grout, R. W., Ma, K.-L., and Chen, J. H. Analyzing information transfer in time-varying multivariate data. In Proceedings of IEEE Pacific Visualization Symposium (March 2011), pp. 99–106.

[141] Wang, C., Yu, H., and Ma, K.-L. Importance-driven time-varying data visualization. IEEE Transactions on Visualization and Computer Graphics 14, 6 (2008), 1547–1554.

[142] Wang, C., Yu, H., and Ma, K.-L. Application-driven compression for visualizing large-scale time-varying data. IEEE Comput. Graph. Appl. 30 (January 2010), 59–69.

[143] Wang, W., Lu, A., Yu, L., and Li., Z. A digital lego set and exercises for teaching security protocols. In Proceedings of Colloquium for Information Systems Security Education (CISSE) (2008), pp. 26–33.

[144] Ward, M. O., and Guo, Z. Visual exploration of time-series data with shape space projections. Computer Graphics Forum 30, 3 (2011), 701–710.

[145] Weaver, R. J., and Luettich, R. A. 2d vs. 3d storm surge sensitivity in adcirc: Case study of hurricane isabel. In Estuarine and Coastal Modeling XI (2010).

[146] Weinberg, J., Engel, G., Gu, K., Karacal, C., Smith, S., White, W., and Yu, X. A multidisplinary model for using robotics in engineering education. In Proc. of The American Society for Engineering Education Annual Conference (2001).

[147] Weinberg, J., and Yu, X. Robotics in education: Low-cost platforms for teaching integrated systems. IEEE Robotics and Automation 10, 2 (2003), 4–6.

[148] Wohlfart, M., and Hauser, H. Story telling for presentation in volume visualization. In Proceedings of EuroVis (2007), pp. 91–98.

[149] Wold, S., Esbensen, K., and Geladi, P. Principal component analysis. Chemometric and intelligent Lab. Sys. 2 (1987), 37–52.

[150] Woodring, J., and Shen, H.-W. Incorporating highlighting animations into static visualizations. In Proceedings of SPIE Electronic Imaging (2007).

[151] Woodring, J., and Shen, H.-W. Multi-scale time activity data exploration via temporal clustering visualization spreadsheet. IEEE Transactions on Visualization and Computer Graphics 15, 1 (Jan 2009), 123–137.

[152] Woodring, J., and Shen, H.-W. Semi-automatic time-series transfer functions via temporal clustering and sequencing. Computer Graphics Forum 28, 3 (June 2009), 791–198.

[153] Woodring, J., Wang, C., and Shen, H.-W. High dimensional direct rendering of time-varying volumes. In Proceedings of IEEE Visualization (2003), pp. 417–424.

[154] Wyeth, P., and Purchase, H. Using developmental theories to inform the design of technology for children. In Proc. of Conf. on Interaction design and children (2003), pp. 93–100.

[155] Xie, Z., Huang, S., Ward, M., and Rundensteiner, E. Exploratory visualization of multivariate data with variable quality. In IEEE Symposium on Visual Analytics Science and Technology (2006), pp. 183–190.

[156] Yu, H., Wang, C., Grout, R. W., Chen, J. H., and Ma, K.-L. In situ visualization for large-scale combustion simulations. IEEE Comput. Graph. Appl. 30 (May 2010), 45–57.

[157] Yu, L., Harrison, L., Lu, A., Li, Z., and Wang, W. 3d digital legos for teaching security protocols. TLT 4, 2 (2011), 125–137.

[158] Yu, L., Lu, A., and Chen, W. Generating time lines with virtual words for time-varying data visualization. International Symposium on Visual Information Communication and Interaction(VINCI) (2012).

[159] Yu, L., Lu, A., Ribarsky, W., and Chen, W. Automatic animation for time-varying data visualization. Computer Graphics Forum 29, 7 (2010), 2271–2280.

[160] Yuan, X., Xiao, H., Guo, H., Guo, P., Kendall, W., Huang, J., and Zhang, Y. Scalable multi-variate analytics of seismic and satellite-based observational data. Visualization and Computer Graphics, IEEE Transactions on 16, 6 (nov.-dec. 2010), 1413 –1420.

[161] Zhao, W., Chellappa, R., Phillips, P. J., and Rosenfeld, A. Face recognition: A literature survey. ACM Comput. Surv. 35, 4 (2003), 399–458.

[162] Zijlema, M. Computation of wind-wave spectra in coastal waters with swan on unstructured grids. Coastal Engineering 57, 3 (201), 267 – 277.