VIEW-DEPENDENT VISUALIZATION FOR ANALYSIS OF LARGE DATASETS

A Dissertation

by

DEREK ROBERT OVERBY

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

December 2011

Major Subject: Computer Science

VIEW-DEPENDENT VISUALIZATION FOR ANALYSIS OF LARGE DATASETS

A Dissertation

by

DEREK ROBERT OVERBY

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Approved by:

Chair of Committee,	John Keyser
Committee Members,	Selma Childs
	Frank Shipman
	Jim Wall
	Vinod Srinivasan
Head of Department,	Duncan Walker

December 2011

Major Subject: Computer Science

ABSTRACT

View-Dependent Visualization for Analysis of Large Datasets. (December 2011)

Derek Robert Overby, B.S., Texas A&M University;

M.S., Texas A&M University

Chair of Advisory Committee: Dr. John Keyser

Due to the impressive capabilities of human visual processing, interactive visualization methods have become essential tools for scientists to explore and analyze large, complex datasets. However, traditional approaches do not account for the increased size or latency of data retrieval when interacting with these often remote datasets. In this dissertation, I discuss two novel design paradigms, based on accepted models of the information visualization process and graphics hardware pipeline, that are appropriate for interactive visualization of large remote datasets. In particular, I discuss novel solutions aimed at improving the performance of interactive visualization systems when working with large numeric datasets and large terrain (elevation and imagery) datasets by using data reduction and asynchronous retrieval of view-prioritized data, respectively.

First I present a modified version of the standard information visualization model that accounts for the challenges presented by interacting with large, remote datasets. I also provide the details of a software framework implemented using this model and discuss several different visualization applications developed within this framework.

Next I present a novel technique for leveraging the hardware graphics pipeline to provide asynchronous, view-prioritized data retrieval to support interactive visualization of remote terrain data. I provide the results of statistical analysis of performance metrics to demonstrate the effectiveness of this approach.

Finally I present the details of two novel visualization techniques, and the results of evaluating these systems using controlled user studies and expert evaluation. The results of these qualitative and quantitative evaluation mechanisms demonstrate improved visual analysis task performance for large numeric datasets.

To my parents, for their love and support, and my children, who inspire and motivate me.

ACKNOWLEDGMENTS

I would like to acknowledge all my friends and colleagues, who have inspired and assisted with portions of this work. Contributions by Blake Stoker, Eric Landreneau, Larkin O'hern, and Abhishek Bhor are especially appreciated.

NOMENCLATURE

bpp	bits per pixel
CADRG	Compressed Arc Digital Raster Graphics
CC	COP Currency
CIB	Controlled Image Base
CMV	coordinated multiple view
COP	Common Operating Picture
DTED	Digital Terrain Elevation Data
EMS	electro-magnetic spectrum
FBO	frame buffer object
GEF	Graphical Editing Framework
GPGPU	general-purpose GPU
GPU	graphics processing unit
HSCB	human social-cultural behavior
LOD	level-of-detail
MCR	Message Completion Rate
MILCs	Multi-dimensional In-depth Long-term Case studies
NGA	National Geospatial-Intelligence Agency

ProDV	Process-Oriented Data Visualization
RPF	Raster Product Format
RTCA	real-time casualty assessment
SA	situational awareness
SAGAT	Situational Awareness Global Assessment Technique
SART	Situational Awareness Rating Technique
SDK	software development kit
SoS	Speed-of-Service
STV	Spatio-Temporal Visualizer
TLM	topographic line map
TSA	team situational awareness
VDPM	view-dependent progressive mesh
VTK	Visualization Toolkit

TABLE OF CONTENTS

LIST OF FIGURES

FIGURE Page

FIGURE Page

FIGURE Page

LIST OF TABLES

CHAPTER I

INTRODUCTION

The visualization and analysis of large, complex datasets is a challenge for many researchers and scientists. Due to the impressive capabilities of human visual processing, interactive visualization methods have become essential tools for scientists to explore and analyze these datasets. However, traditional approaches to implementing visualization software are often not designed to deal effectively with either the size or latency of data retrieval when interacting with large (often remote) datasets. And while reference models for the design of visualization systems do exist, these models do not account for dealing with large remote datasets that have become prevalent in so many scientific domains. In this dissertation I will discuss two novel design paradigms, based on accepted models of the information visualization process and graphics hardware pipeline, that are appropriate for interactive visualization of large remote datasets. In particular, I discuss novel solutions aimed at improving the performance of interactive visualization systems when working with large numeric datasets and large terrain (elevation and imagery) datasets by using data reduction and asynchronous retrieval of view-prioritized data, respectively. I also present the details of three visualization applications implemented using the proposed techniques, and the results of evaluating these applications using controlled user studies, statistical analysis of performance metrics, and expert evaluation.

A. Problem Statement

In many government, academic, and industrial domains today the task of extracting timely and relevant information from large datasets poses great challenges to many applications.

The journal model is *IEEE Transactions on Visualization and Computer Graphics.*

In the military community large datasets are routinely collected during testing and training events involving both simulation and sensor data collected over long periods of time, in addition to large amounts of network data. However many currently used visualization applications do not account for the need to interact with large, remote datasets. Therefore novel techniques that improve interaction when working with such datasets are needed.

While traditional methods can provide us the ability to visualize these datasets, the ability to maintain interactive rates for direct manipulation interactions using standard visualization methods often decreases as the size of the dataset increases. Also, the latency of retrieving the data may be unknown and may exhibit a large degree of variation, depending on the network environment. I this dissertation I explore modifications to existing visualization models aimed at optimizing visualization implementations when dealing with two specific classes of large datasets. This exploration is important because such models are used in the design and development of many modern visualization systems. I will focus my effort on the exploration of visualization system design within two distinct contexts that I have encountered in my work: large [several gigabytes (GB)] numeric datasets (often empirical data collected by electronic sensors) and extremely large [several terabytes (TB)] terrain imagery datasets, which are often by necessity stored remotely. Each visualization can usually optimize storage of its required data in a data structure that is significantly smaller than the source dataset. Therefore in this work I will discuss a modified design model that will enable multiple complex visualization techniques to be used by domain experts in the analysis of different types of numeric data. This approach allows the data reduction process to be specified by the user using an interactive dataflow model, and multiple interactive visualization techniques to be applied by efficiently segmenting reduced data. Second, when working with extremely large remote datasets (as is often the case with high-resolution terrain and imagery data), view-based prioritization of the retrieval of data is important to optimize both the visual representation of the data and the use of limited local memory resources. Therefore I also propose and explore a technique that allows the modern graphics pipeline (often provided by accelerated graphics hardware) to easily view-prioritize remote data interactively and asynchronously load large amounts of terrain imagery.

B. Approach

My discussion of this research is organized into three main branches, corresponding to Chapters III–V. First, I will discuss a proposed framework for interactive visualization applications that allows visualization techniques to be applied to large datasets by requiring a data reduction model to be specified so that visualization algorithms can operate interactively using minimized blocks of numeric data. I will also describe and discuss how this proposed framework is implemented in an interactive visual programming environment that has been used to develop custom interactive visualization applications in a variety of domains. I will next describe the design of an interactive terrain visualization application that is optimized by efficiently computing a view-dependent prioritization of remote data. By loading data asynchronously, interactive exploration is not hindered by dataset size or latency. Finally, in order to demonstrate the success of practical applications using these techniques, I present the result of developing two novel interactive visualization techniques implemented with the proposed visualization framework, as well as user evaluation results and expert opinion.

1. A Framework for Interactive Visualization Systems

Researchers in the Visual Analytics community have developed custom software applications for a variety of domains. In this section I will discuss work performed in conjunction with a U.S. government customer. The goal of this effort was to provide useful visual analytics capabilities to several distinct groups of data analysts. I developed a novel adaptation of the information visualization model that improves the performance and flexibility of custom interactive visualization applications. I will show how the implementation framework based on this model, which includes an interactive visual programming environment, was used to develop several successful visualization applications. I will discuss the design of our approach, the observed results of experts using our software in several analysis cases, the development of a user-specified visual analysis requirements, and results of this work using multiple evaluation methodologies.

2. View-Prioritized Data Retrieval

Many efforts in science and technology have collected and archived massive amounts of data. While commercial entities have certainly collected vast amounts of imagery via satellite, certain spatial analysis tasks require much higher-resolution terrain topology and imagery. Analysis of near-earth RF propagation models is but one example; others exist within the domains of disaster recovery, military operations planning, and excavation operations. The introduction of commercially-available sensors that can provide such high-resolution data for a small area, as well as increased commercial satellite imagery and terrain resolution for certain areas makes this work particularly relevant. In this work I use high-resolution terrain data from the National Geospatial-Intelligence Agency (NGA) to support scientific analysis of various geospatial datasets. I will discuss use of the highest resolution data openly available to government agencies in the form of Digital Terrain Elevation Data (DTED) and Controlled Image Base (CIB) satellite imagery.

Previous work in the development of view-dependent visualization methods has provided a wide variety of level-of-detail mesh simplification algorithms designed to reduce the overall count of input geometry to the graphics processing pipeline. These methods have been employed to implement effective interactive visualizations of large 3D environments and CAD models containing large numbers of polygons. Prior work has provided solutions to many challenges to this approach, including silhouette preservation [44], and imperceptible simplification [46]. However, scientific visualization methods usually seek to eliminate visual simplification methods whenever possible to ensure the highest accuracy possible in the visual representation of scientific data. Other recent methods have provided efficient GPU-based terrain rendering implementations [75, 14]; however, these methods do not account for the use of a remote terrain database that must be queried at runtime, and require significant pre-processing of data.

I present a new approach that defines a strategy to efficiently allocate and index segments of video memory for asynchronous retrieval of texture data at full resolution in order to best support accurate scientific visual analysis. This method allows the visualization application to efficiently allocate video memory, retrieve the highest-priority data (defined by the interactive view), and load the data asynchronously while minimizing the impact to the main rendering thread. I will demonstrate the efficiency of this method by conducting a statistical analysis of performance metrics collected from the visualization implementation based on the proposed technique.

3. Applications

I will discuss two interactive visualization applications implemented using the proposed visualization software framework. The first visualization application is designed to enable interactive analysis of temporal patterns in spatial sensor activity. The second visualization application is designed to enable interactive analysis of situational awareness data in a distributed system. The success of these custom visualization applications also serve to demonstrate the effectiveness of the proposed visualization framework.

a. Glyph Visualization for Interactive Analysis of Temporal Data

As distributed sensor networks become more common, methods that enable quick recognition and proficient analysis of pattern irregularities from these sensors will become more important. This portion of my work aims to improve visual recognition of temporal patterns within a network of spatial sensors.

This technique can be applied to a wide range of applications, but is inspired by interaction with colleagues who collect and analyze large datasets produced by networks of spatial sensors. The primary challenge was to provide a quick and thorough overview of daily recorded sensor activity with emphasis on highlighting anomalous behavior. During initial interviews with domain experts, several specific analytical tasks were identified: users should be able to quickly and accurately identify failure patterns within the sensor network and investigate the temporal and spatial context of these failures; users should be able to easily recognize expected patterns such as coordinated movements; users should be able to identify and investigate the context of spatial reporting anomalies to support analysis of sensor problems. I will discuss the design and development of the visualization technique with these user-defined analysis goals in mind. Because it is difficult to compare the use of this method to existing techniques, I will present evaluation of this work using a series of small user studies, as well as collecting expert opinion from users within the application domain.

b. Interactive Analysis of Situational Awareness Data

Situational awareness is made up of the information a person currently has available about his surroundings and relevant peers [18]. Situational awareness data is often provided using electronic equipment designed to display real-time data from networks of spatial sensors. Because this information is usually delivered over a wireless network in a rapid decision-making environment, the latency of information delivery over the network can have a direct effect on human situational awareness and therefore decision-making. To be more specific, situational awareness data that is inaccurate can have a negative impact on decision-making involving coordinated activities if, for example, a user's perception of his peers' locations is incorrect.

In working with analysts attempting to evaluate and compare the performance of various digital systems designed to provide situational awareness (SA) data, I proposed a set of custom visual metrics to support overview and detailed performance analysis of the data collected during test exercises. These exercises were designed to test actual users of the equipment in real-world scenarios involving coordinated activities so that the accuracy of different individuals' SA data can be compared and analyzed (e.g., comparison of team leader vs. team member during convoy). The goal of this work was to compute the proposed metrics and provide interactive software to analyze the impact of network latency at specific times relating to decision-making efforts in the exercise.

C. Summary

To summarize the contributions I will present in this work:

- I propose a novel methodology to ensure interactive performance of visualization systems designed to enable visual analysis of large numeric datasets.
- I propose a novel technique to improve interactive three-dimensional rendering of high-resolution terrain data designed to support accurate visual analysis of scientific data.
- I propose a novel visualization technique to improve analysis of temporal patterns exhibited by a network of spatial sensors.

- I propose a novel metric and visualization technique to improve performance analysis of digital situational awareness systems.

All of these contributions focus on improving interactive visual analysis of large datasets using computational visualization methods. These contributions are necessary to ensure that new and existing visualization techniques can be effectively applied to large, remote datasets that are common in many research domains.

CHAPTER II

BACKGROUND

There are several areas within the broader field of Computer Graphics that I will address as background work for this dissertation. First, the majority of this work has been heavily influenced by previous efforts in the distinct but related fields of Information Visualization and Visual Analytics. I will begin by reviewing seminal and related works in these areas, with specific focus given to the visualization reference models that have been contributed by these communities. Second, because part of this work focuses on specific implementations of interactive visualization applications, I will discuss the major relevant works contributed from several segments of the broader visualization community that focus on glyph-based visualization techniques and interactive visualization of temporal data. This is important because while distinct visualization applications are rarely similar enough for direct comparison, it is helpful to understand the origins and limitations of previous approaches that are often combined to create new techniques. Also, I will discuss some existing visualization applications that are similar in specific aspects to the approach presented here. I will review major contributions to techniques for optimization of terrain visualization applications such as level-of-detail (LOD) simplification algorithms. I will provide a brief overview of the recent evolution of graphics processing unit (GPU) hardware, which due to its programmability, high degree of parallelism, and optimized hardware implementations, has enabled the development of many hardware-accelerated visualization and computational techniques. Finally, I will discuss prior work in defining and evaluating situational awareness (SA), which is relevant to a specific analytical technique presented in this work.

A. Information Visualization

In this section I will discuss relevant prior work in Information Visualization, specifically the primary reference models and evaluation techniques that have been contributed by researchers in this community.

1. Reference Models

Abstractions of the information visualization process are often evaluated in terms of the visualization reference model, originally proposed by Card et al. [8], that classifies human interactions with visualization systems as one of several classes: data transformations, visual mappings, or view transformations. This seminal model, depicted in Figure 1, has been referenced heavily within the information visualization community. The primary contribution of the information visualization reference model, as it has come to be known, has been the standardized categorization of novel interaction techniques into these three major classes. I will appropriately describe the contributions of this work in reference to this model, with modifications.

Around the same time that the information visualization reference model was proposed, another model that focuses on the multiple states of data throughout the visualization and interactive analysis process was presented by Chi [10]. This model has been used extensively in this community as a taxonomy by which to characterize the state of data and the transformations applied to data at each state by users of interactive visualization applications. A simplified depiction of this model, called the *data state model*, is shown in Figure 2. The data state model has been used to taxonomize (and therefore provide some grounds for comparison) of visualization techniques according to the state of data that they interface with. Both of these visualization reference models have been influential in the design and development of interactive visualization software.

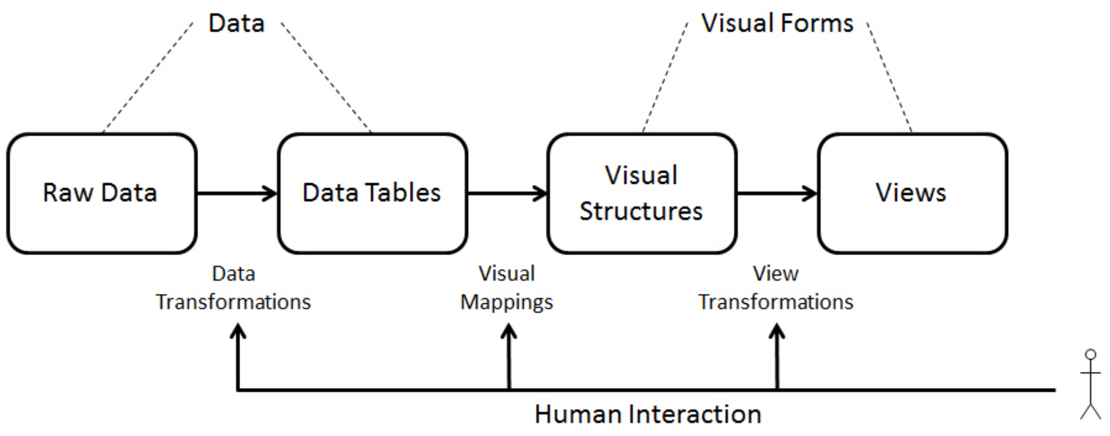


Fig. 1. A depiction of the information visualization reference model, originally proposed by Jock Mackinlay, Stuart Card, and Ben Shneiderman [8]. This model has been widely referenced in the visualization research, and is often used to aid in the categorization of visualization techniques as *data transformations*, *visual mapping* operations, or *view transformations*.

Several notable software toolkits have also contributed practical models of visualization components, including the well-know Visualization Toolkit (VTK) [63], and prefuse (intentionally not capitalized by its authors) [30]. prefuse provides a Java-based library of popular information visualization techniques such as fish-eye lenses, which are used to provide user-controlled scaling of two-dimensional information renderings, and hyperbolic trees, which are used to optimize layout of large hierarchical structures for visual display. The broader goal of information visualization and visual analytics research is to facilitate understanding of how interactive software can better enable users of visualization software to perform complex analytical tasks.

Recent work has explored collaboration impacts the design and implementation of methods meant to aid in visual analysis of large collections of data. For example, two applications with obvious need for collaboration are visual analysis of astrophysics and intelligence data. Work by Heer and Agrawala [29] has sought to formalize

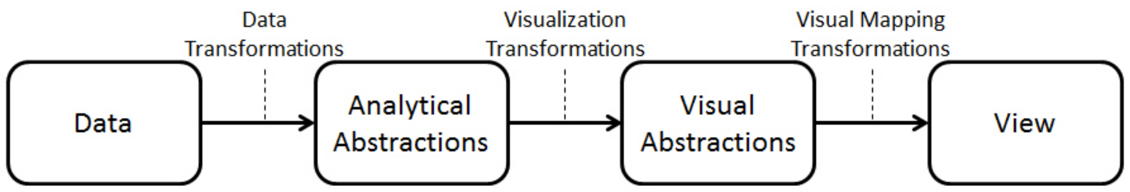


Fig. 2. A simplified depiction of the data state model, originally proposed by ed Chi [10]. This model is often used to taxonomize visualization techniques by the state of data that they interface with.

various approaches to this issue and identify numerous important features required to support collaborative analysis in interactive visualization environments. Evaluation of the effectiveness of different approaches to support collaborative analysis has seen very little attention, however.

2. Evaluation of Visualization Techniques

There have been many case studies that demonstrate how particular tools have been successful in enabling domain experts to more effectively explore and analyze data. One recent example is the work of Tesone and Goodall, where a coordinated multiple view (CMV) visualization was applied to analysis activities within the information assurance (IA) domain [67]. Lam et al discussed the benefits gained by analysts of large collections of web search activity using their Session Viewer software [38]. Exactly how to quantitatively measure the contribution of such work is not always clear; and therefore not many recent studies such as these have included verifiable, experimental results validating their effectiveness.

Recently, Shneiderman and Plaisant have provided much needed guidance on the way forward for evaluating the effectiveness of domain-specific visual analytics applications [65]. They suggest that Multi-dimensional In-depth Long-term Case studies (MILCs) can be used to evaluate the success of information visualization and visual analytics research by qualitatively measuring the success of specific analytic goals self-reported by expert users over long periods.

The results of these works suggest that reported success of domain experts attempting specific analytic goals within their own operational environments might provide additional insight into the utility of a specific approach, compared with traditional task-based user studies often conducted with students instead of domain experts. Also, self-reported analytic insights over more generous time periods can be used to more effectively determine how well domain experts are able to integrate a new technique into an existing analysis process.

3. Visual Analytics

The recently-emerged field of Visual Analytics is defined as "the science of analytical reasoning facilitated by interactive visual techniques" [13]. The goals of this research are enumerated in a seminal publication edited by the late Jim Thomas, among others. Some of the primary challenges enumerated by this research agenda include supporting the process of human decision-making by enabling users to "detect the expected and discover the unexpected" in large, complex datasets composed of heterogeneous and often conflicting data. Work in this field has become particularly relevant due to the prevalence of situations in many domains that require human decision-makers to rapidly ingest information from large collections of data to make critical decisions in emergencies. Researchers in this field have contributed many novel visual analysis techniques in domains such as computer security (sometimes referred to as cyber-security) [34] [47] and biological studies [2].

B. Visualization Techniques

In this section we briefly review related work in glyph visualization techniques and interactive visual analysis methods focused on interactive display of data for temporal pattern recognition, as well as existing visualization applications often used by developers and end-users to perform analytical tasks.

1. Glyph Visualization

Within the fields of scientific and information visualization, there has been significant exploration of the use of iconic-based representations of information. Often these visualization techniques seek to simplify visual display of multi-variate data by aggregating multiple attributes into a glyph representation. For example, a circular glyph can be used to represent a data element on a map. Obviously the center of the circle is used to represent the object's location, but the radius, color, texture, outline, or any other visual attributes can be used to represent other dimensions of data. An example of this is work by Ribarsky that explored user-specified glyph generation in the context of multivariate spatial data [61]. Some of the basic concepts used to map glyph properties to data attributes were formalized by Post et al. [59]. Based on this, one of the visualization techniques I will present maps temporal data attributes to the scale and texture of a glyph, while glyph shape is manipulated to depict temporal aggregations. Ebert et al. have published results on using procedural generation of glyphs to aid in the analysis of various multivariate datasets, including document collections and fluid flow [16]. Glyphs have also been applied to flow visualization, as in an application of glyph-based visualization of turbulence within incompressible flow by Kirby et al. [36]. Other research has focused on addressing additional challenges with perception or other domain-specific analysis issues, including uncertainty [73]. My work builds on these previous efforts by exploring a custom glyph designed to facilitate rapid visual inspection of temporal activity patterns exhibited by spatial sensors.

2. Temporal Visualization

Considerable work has been done examining different visual representations and interaction methods to support visual analysis of temporal data. By temporal data, I mean data that has a temporal value for each record (where data is, in general, a sequence of records). Because of the large variety of domains in which custom temporal visualization techniques have been developed, I will focus on seminal work in this area and interactive temporal visualization techniques that are similar in application to the work presented in this dissertation. Much of this work has focused on developing techniques to support domain-specific analytical goals. Early work by Plaisant et al. explored using timelines and glyphs to represent temporal events within a patient's medical history [58] in order to improve recognition of specific patterns of past medical events when performing medical diagnosis. More recent work by Wang et al. [71] evaluated a custom glyph-based visualization technique used to infer causal relationships between events in electronic health records. In the network security domain, Keim's CircleView technique was developed to aid in the visual analysis of large amounts of dynamic temporal data using intuitive circular graphs [35]. Another method, Spiral, focused on highlighting periodic patterns in temporal data, such as the seasonal behavior patterns of primates [9]. Weaver et al. employed several visualization techniques to explore recurring temporal event patterns within hotel visitation records [72].

As the average size of real-world datasets have continued to increase, researchers have explored various approaches to enhance the capabilities of visual interfaces accordingly. Research exploring interaction mechanisms for temporal analysis of large molecular biology datasets using interactive visual queries has been contributed by Hochheiser [31]. Other work examined use of aggregation to facilitate temporal analysis of large collections of public safety data [26]. A custom visualization technique presented in this dissertation makes similar use of aggregation to reduce overplotting when visualizing large amount of data at various scales. Recent work by Bak et al. proposes the use of *growth ring maps* to display temporal and spatial attributes of data to aid in the analysis of a large volume of mouse movement data during experiments [2]. While this approach was shown to be quite effective at enabling scientists to track which areas of the experimental setup were most visited by the mice and behavioral aspects related to a disease could be discerned, this approach was not directly applicable to my work because the sequence of occurrence is not indicated using this technique.

Finally, a comparison by Aigner et al. of several common techniques for analyzing temporal data indicated that event-based visualization techniques could be used to emphasize relevant information within large datasets [1]. A task-based evaluation of a specific application, the Spatio-Temporal Visualizer (STV), has been presented by Chung et al. [11]. This application was designed to allow law enforcement experts to identify crime patterns within spatial and temporal dimensions of historical data.

The techniques presented here build on the core concepts of temporal visualization to produce a novel visualization technique for a domain-specific analysis task. The details of this novel technique are presented in Chapter V.

3. Existing Visualization Systems

There are some existing visualization systems that are available that employ similar techniques to the work presented here or provide similar capabilities. Although it is not appropriate to make direct comparison between visualization implementations designed for different applications, we note that none of the existing systems provided integrated data caching capabilities necessary for improving interactive visualization performance when dealing with large datasets.

- VisIt is a free visualization tool and software development kit (SDK) designed by Lawrence Livermore National Laboratory for parallel visualization applications of scientific data [39]. However, while VisIt was designed to handle very large datasets and provides capabilities for rapidly developing custom visualizations and interfaces, it is primarily a visualization developer tool, not an end-user tool appropriate for data analysts.

- LabView is a visualization toolkit that provides a visual programming environment that does provide end-users the capability to configure interactive visualization applications using a variety of data processing and visualization functions [51].

- prefuse (the title is intentionally not capitalized) is a Java-based visualization software development kit that provides several popular interactive information visualization techniques, such as hyperbolic trees, fish-eye lenses, and force-directed node graphs [6]. While prefuse does provide interactive visualization techniques that can be integrated into other visualization applications, it does not provide any data reduction capabilities.

- The Visualization Toolkit (VTK) is another popular software development kit that provides many functions that are useful in the creation of interactive visualization applications [37]. Like prefuse and VisIt, however, VTK does not provide an end-user interface and is designed primarily to support developers of visualization software.

- RapidMiner is an open-source end-user interactive visualization tool that provides interactive data mining capabilities [60]. However while RapidMiner also provides some visualization capabilities, it does not have the capability to develop and integrate custom visualization capabilities.

- KNIME, pronounced [naim], is a data analysis and visualization suite developed at the University of Konstantz that provides a visual programming environment for end-users and is integrated with the Java-based data mining package, Weka [68].
- KnowledgeFlow, developed at the University of Leiden in the Netherlands, also provides a visual interface to the data mining and analysis capabilities provided by Weka, and provides a few interactive visualization features but does not allow integration of custom visualization techniques [40].

C. Terrain Rendering

Interactive terrain rendering, which is often a major component of immersive three-dimensional environments, has seen much attention in the visualization community. Many approaches to building efficient terrain rendering applications have focused on employing LOD mesh simplification algorithms due to the fact that, in most cases, terrain geometry can be represented as a connected mesh of elevation points. In many cases, terrain geometry is organized in a regular grid structure as a result of the data collection mechanisms used to sense elevation along the physical surface of the Earth, primarily satellite-based sensors. Also, terrain databases that store this mesh data are often large if they cover a significant area or are high-resolution (for the purpose of this work, a terrain mesh consisting of elevation postings at a 30-meter interval would be considered high-resolution). Therefore, terrain rendering applications must efficiently retrieve data from a remote database at runtime. In this section I will review relevant prior work in this area.

Previous work in the development of terrain visualization methods has provided a wide variety of LOD mesh simplification algorithms designed to simplify the terrain geometry (i.e., reduce the number of vertices) input to the graphics processing pipeline. These methods have been employed to implement effective interactive visualizations of large 3D environments and CAD models containing large numbers of polygons. Early work by Lindstrom et al. proposed an algorithm for rendering continuous LOD terrain meshes by projecting *delta values* for each potential merge operation on the original mesh into screen space and comparing to a threshold [41]. Later work by Lindstrom added efficient memory management approaches and asynchronous refinement [42]. Duchaineau et al. presented the Real-time Optimally Adapting Meshes (ROAM) approach that introduced incremental triangle stripping (which optimizes hardware rendering performance) and priority-based simplification of terrain organized into binary trees [15].

A very similar model was also proposed by Evans et al. that employs right-triangulated irregular networks to simplify terrain meshes [23]. While both approaches use similar simplification techniques, the author claims that RTINs are potentially more memory efficient since they store more triangle data than actual vertex data. There have been several incremental refinements to these initial approached to rendering large terrain models [57], but in general the *view-dependent progressive mesh* (VDPM) approach employed by all these techniques has become widely accepted. Hugues Hoppe also extended such initial work by defining *geomorphs* to introduce temporal continuity when interactively adjusting mesh level-of-detail (i.e., reducing visible ’popping’ of the mesh) [33].

As the accuracy of spatial sensors have increased over the past decade, more recent efforts have by necessity focused on quantifying the trade-offs in memory usage and mesh approximation error [3] and optimizing memory clustering algorithms [4] when using these existing techniques. As I will discuss later, compared to all these existing techniques the approach I will present differs in that the goal is to efficiently retrieve the highest resolution texture data available and render this along a full-resolution mesh surface. Certainly any of these previous techniques can be integrated with our method to locally simplify the mesh in order to improve interactive rendering performance; however, due to the drastically increased performance of modern graphics hardware this may not be necessary.

Additional prior work has provided solutions to many challenges to the LOD approach to terrain rendering, including silhouette preservation [44], and imperceptible simplification [46]. Both of these techniques are aimed at reducing the visible error in a simplified terrain mesh by measuring and minimizing the unavoidable difference between a full-resolution mesh and a simplified mesh. Other recent methods have provided efficient GPU-based terrain rendering implementations [14, 75], however these methods do not account for the use of a remote terrain database that must be queried at runtime.

D. Graphics Hardware

Recent advances in graphics hardware implementations and the effects of these advances in performance and functionality on the community's model of the graphics pipeline are certainly relevant to this work. This is because I will present a novel approach that leverages the advantages of the programmable graphics pipeline to efficiently prioritize the retrieval of high-resolution data from a remote database during interactive rendering. The implementation of this approach requires an understanding of the architecture and capabilities of the modern graphics hardware pipeline.

Initially, hardware implementations of the graphics pipeline were fixed-function. That is, each major component of the traditional graphics pipeline, shown in Figure 3, was defined statically and could not be altered by developers. However, the introduction of programmable vertex and fragment processors allowed developers of visualization applications to begin experimenting with dynamic implementations of these components. This enabled developers to create interactive hardware-accelerated implementations of non-traditional surface shading models that previously could only be implemented using offline rendering applications (such as Renderman). Later, the increased processing power provided by the stream processing model of many-core GPUs enabled researchers to begin

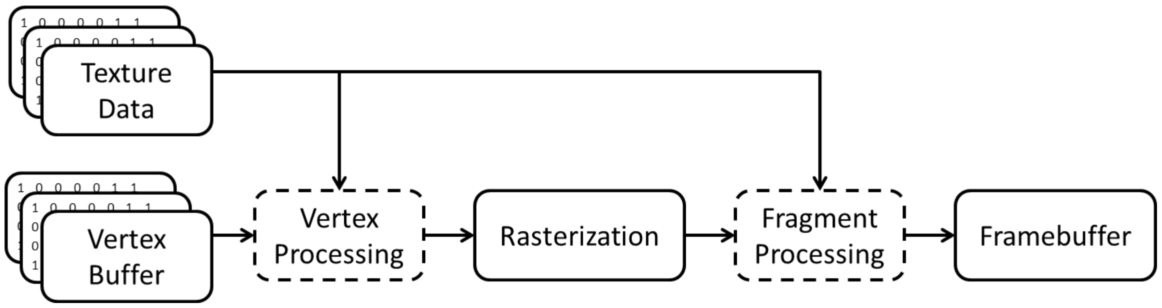


Fig. 3. The major traditional components of a fixed-function graphics pipeline. The programmable components of the pipeline are shown with dashed outlines. The framebuffer output shown in the figure is what is shown on the user's monitor.

exploring how traditional and non-traditional rendering techniques could be parallelized and therefore optimized for such hardware [56]. Effort has certainly been made to leverage this programmability to increase performance and visual quality of scientific visualization and volume rendering applications, since scientific datasets often exhibit direct two- or three-dimensional spatial mappings and are therefore well suited for GPU implementations [22]. Researchers in the information visualization community [49] have also explored how the advantages offered by modern graphics hardware can be used to improve the interactivity of techniques such as parallel coordinates and scatter plots [25], and multi-value data glyphs [17].

This increased processing power also led to the exploration of general-purpose GPU (GPGPU) applications, designed to increase the efficiency of non-rendering algorithms using the stream model [45]. Some have argued that according to Moore's Law, GPU-based implementations of parallelizable algorithms have become more cost-effective than CPU implementations [56].

Several programming languages dedicated to the programmable graphics pipeline have matured over the past decade. These include Cg from nVidia [48], GLSL [62], and

Microsoft's HLSL [53], each of which have similar base capabilities but differ slightly in syntax and implementation of high-level functions. In addition to these languages, which are focused on rendering applications, other languages have also been developed to enable GPGPU applications not designed to produce visual output to the framebuffer but instead produce numerical results. Most notably these include CUDA from nVidia [52], the Stream Computing SDK from ATI [5], and Brook from Stanford [7].

E. Situational Awareness

Finally, in this section I will review seminal work in the area of situational awareness, since prior work in this area is relevant to a particular visual analysis application presented in this dissertation.

Situational awareness in the context of decision-making operations is defined as "what information is important for a particular job or goal" [19]. Situational awareness, while a necessary aspect of any coordinated operation, is difficult to quantify and therefore analyze for impact on decision-making. Initial contributions by Endsley included a model of situational awareness consisting of three levels: perception, comprehension, and projection [18]. For the purpose of this work, we will be primarily focused on the accuracy of data that is presented to the user during SA Level 1 (perception).

Work by Endsley et al. goes on to define team situational awareness (TSA) [18], as well as qualitative evaluation mechanisms for such [20]. The study of situational awareness has roots in these human cognition studies by Endsley et al. [21] in which the visual attention and resulting situational awareness of air traffic controllers was measured by questionnaires designed to evaluate the operator's mental model at specific times during a defined process—perception, comprehension, and projection [18]. Other research efforts in visualization applications for improving SA have examined the use of various image generation techniques (such as “blobby shading” and use of Delaunay triangulation) to aggregate complex and cluttered visual features to improve SA [32]. Still other efforts have focused more on the “theater-wide situational awareness covering a large geographic area containing many military units” [24], as opposed to this work that focuses on measuring the accuracy of SA data locally for each participant. A key novel contribution of this work is a quantitative metric that provides insight into the analysis of SA and therefore decision-making.

There have also been efforts to provide interactive visualization techniques that enable situational awareness within various contexts, such as emergency response, network intrusion detection, and biosurveillance [43]. These techniques often make use of qualitative SA metrics such as the Situational Awareness Global Assessment Technique (SAGAT) and the Situational Awareness Rating Technique (SART), both of which rely on both objective and subjective questionnaires, respectively, designed to evaluate a person’s comprehension (SA Level 2) and projection (SA Level 3) of various aspects of the situation at multiple points in a decision-making process [21].

CHAPTER III

A FRAMEWORK FOR INTERACTIVE VISUALIZATION OF LARGE NUMERIC DATASETS

Researchers in the Visual Analytics community have developed custom software applications for a variety of domains in order to support the needs of expert analysts and decision-makers. In this chapter I will discuss work performed in conjunction with a U.S. government customer. The goal of this effort was to provide useful visual analytics capabilities to several distinct groups of domain experts within this community. This approach allowed these capabilities to be delivered in a manner that enables the domain experts to learn and apply the necessary skills in their own environment. I will discuss how we have provided guided training sessions to introduce domain experts to each analysis case presented here, including analysis of network performance, anomalous sensor activity, and engagement results. Through long-term interactions with these experts, we have begun to observe analysts using skills from prior training sessions to propose new analytic requirements. I will discuss the design of this approach, the observed results of experts using the software in several analysis cases, the development of a user-specified visual analysis requirement, and results of this work using three different evaluation methodologies.

A. Approach

The proposed approach to developing a visualization application to work in a dynamic environment containing many types of large datasets is based on the existing model of information visualization initially proposed by Card et al. The existing model, shown earlier in Figure 1, does distinguish the difference between data transformation operations and view transformation operations, however it does not account for performance considerations based on the size of the input data for applications involving large datasets.

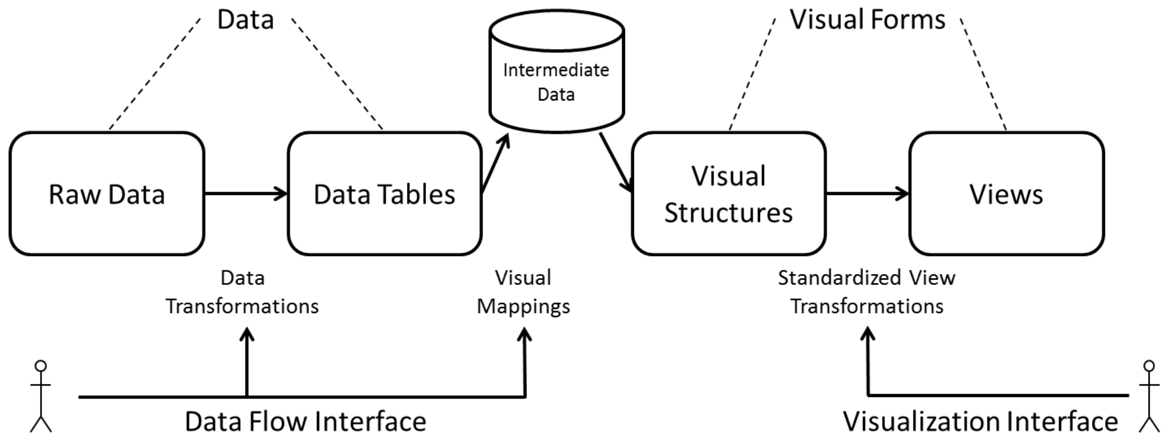


Fig. 4. The proposed framework is based on a modification of the existing information visualization model. In the proposed model, data transformation and reduction operations are configured using one user interface, and interactive view transformation operations are controlled using another visualization interface. In this model data processing and visualization techniques can be developed and configured independently and reduced data is cached to increase performance when working with large datasets.

Therefore, I propose a novel modification to the existing model that ensures that interactive visualization software applications are not overloaded with dense volumes of data that can significantly impact rendering performance. In the proposed model, shown in Figure 4, data transformation and reduction operations are configured using one user interface, and interactive view transformation operations are controlled using another visualization interface. This allows us to standardize and simplify the user interface for each of these tasks independently, which can improve the user experience. But more importantly, in this model data processing and visualization techniques can be developed and configured independently and data is pre-processed, reduced, and cached to increase interactive rendering performance when working with large datasets.

Although it is not appropriate to make direct comparison between visualization implementations designed for different applications, it is necessary to at least provide a capabilities-based comparison with other similar systems. In order to conduct this comparison, it was necessary to first identify the primary system requirements by which each system is evaluated.

Visual programming environment A *visual programming environment* is required to allow users without advanced programming skills to be able to easily create customized visualization applications.

Multiple visualization techniques Users we spoke with were interested in using *multiple types of visualizations* since they were responsible for analysis of multiple data types (e.g., temporal and spatial data).

Coordinated multiple views *Coordinated multiple views* are often provided to enable effective analysis when using multiple visualization techniques.

Custom visualizations The development and integration of *custom visualizations* enables complex data to be explored in new ways. This was quite important as several new visualization capabilities that these users required had already been identified (e.g., sensor performance analysis).

Unified visualization interface A *unified visualization interface* is desirable because this feature simplifies software training for new users, and trained users can apply existing skills to new visualization methods.

Data reduction and caching *Data reduction* is required to ensure that large numeric datasets do not limit the performance of interactive visualization software. This can be accomplished by providing numeric aggregation algorithms or other custom data processing routines.

Table I. Visualization System Capability Comparison

	ProDV	prefuse	VTK	VisIt	LabView	RapidMiner	KMINE	KnowledgeFlow
Visual Programming Environment	Y	N	N	Y	N	Y	Y	Y
Multiple Visualization Methods	Y	Y	Y	Y	Y	Y	Y	Y
Coordinated Views	Y	N	N	Y	N	N	Y	N
Custom Visualizations	Y	Y	Y	Y	Y	N	Y	N
Unified Visualization Interface	Y	N	N	N	N	N	N	N
Data Reduction	Y	N	N	Y	N	Y	Y	Y
Data Caching	Y	N	N	N	N	N	N	N
End-User System	Y	N	N	N	N	Y	Y	N

Data caching *Data caching* reduced overall computational cost of working with large datasets, and also enables other useful features such as collaborative analysis.

End-user system And finally, while many systems provide excellent visual analysis capabilities, those that are not designed as *end-user systems* (i.e., systems that do not require advanced programming skills or detailed Computer Science knowledge) are difficult for many users to operate.

The results of the capabilities-based comparison are shown in Table I. While most other systems do provide multiple visualization techniques and some also provide a visual programming environment, few provide support for coordinating multiple views. Also, only two of the other systems are for end-users, whereas the rest are primarily tools for software developers. And most notably, none of the other systems provides mechanisms for both data reduction and caching, which are necessary when building interactive visualizations of large datasets. The implementation of the proposed framework, called Process-Oriented Data Visualization (ProDV), provided all the required features.

1. User Environment

In work with a government client, the primary goal was to provide interactive visualization software to support visual analysis. The user's operating environment consisted of constantly varying forms of data that required integration and visual analysis for both anomaly detection and metric evaluation over long periods. While various commercial packages exist to provide some of the basic visualization capabilities used, the degree of customization required and the government software certification needs made it more advantageous to develop a novel visualization environment. In this section I discuss the approach to delivering flexible visual analytics capabilities to a government customer in a dynamic environment. In later sections I will discuss more specific analysis cases in detail.

The primary user for this work was a U.S. government organization responsible for testing and performance evaluation of new equipment and systems based on predefined requirements. The results of these evaluations are used to support purchasing decisions. Large-scale testing activities often compare performance of equipment from different manufacturers. Each test defines specific performance metrics designed to evaluate the stated requirements. For example, networking equipment might be evaluated for performance using metrics such as Speed-of-Service (SoS) or Message Completion Rate (MCR), but depending on the implementation these metrics might be evaluated at the Application layer for some tests (using software log data) and at the Transport layer for others (using recorded network traffic data). Spatial and temporal sensors are most often evaluated in terms of the number of detected failures or anomalies in a given time.

Although analysis requirements are usually specified explicitly before testing begins, it is common practice in this environment to capture and record all forms of electronic data available over extended periods of time. Therefore many of the datasets they need to perform analysis on are large collections of multimodal data. Sanitized versions of these datasets have been used for this work, and are typically on the order of several gigabytes. In some cases, these users are interested in applying these techniques to unsanitized datasets that may be several hundred gigabytes or more.

2. Requirements

Based on this understanding of the environment, it became clear that any visual analysis capabilities provided should not only provide all the basic information visualization interactions (i.e., zoom and filter, details on demand) [64], but also should be easily customizable by domain experts to account for changing evaluation requirements and input data format or semantics. In particular, the experts need to be able to define and incorporate new visualization or analysis tools themselves to address new evaluation criteria, rather than just use a preselected set of information. Users also need to be able to import data from a wide variety of different sources (e.g., different types of sensors stored in different database formats), and combine these sources into a CMV visualization.

3. Framework

We began by defining a framework architecture that defines common objects and the interfaces between them, which is referred to as the Process-Oriented Data Visualization (ProDV) framework. In order to be effective in the user's operational environment, this framework needed to be based on a simple abstraction that could be represented visually and easily understood by domain experts. The majority of domain experts were quite familiar with the equipment domain (e.g., networking), but were not familiar with computational algorithms or data structures beyond common RDBMS objects. They were accustomed to using common tabular computation tools such as Microsoft Excel.

4. Framework Components

The framework components defined in this abstraction are based on an understanding of the average domain user's conceptual model of the visualization process. We extracted this model from early interviews with data analysts and compared the objects/functions most identified by domain experts with formal reference models that have been defined in the visualization community. Key insights forming the basis of this abstraction included:

- Users were knowledgeable about the semantics and format of the data.
- Users had an idea of how they would like to visualize data (e.g., using a graph or a map).
- Users understood that data needed to be processed and had a vague idea of how that might happen, but did not fully understand any of the specific data structures or algorithms used.
- Users understood that data might need various transformation operations.

From these basic observations gleaned from experience interviewing a range of potential users, we arrived at the framework based on this simple abstraction shown in Figure 5. As shown in the figure, the ProDV framework defines data sources, data operators, processing modules, cached data and visualizations as the primary objects in our abstraction of the visualization process. In terms of the visualization reference model this framework divides the workspaces that transformations are conducted in. Only data transformations and visual mappings are manipulated while interacting with framework components, while only view transformations can be manipulated from within the visualization environment.

Data sources *Data sources* provide tuple records (records with multiple typed fields) from a variety of different sources. SQL databases, structured text files, and local database

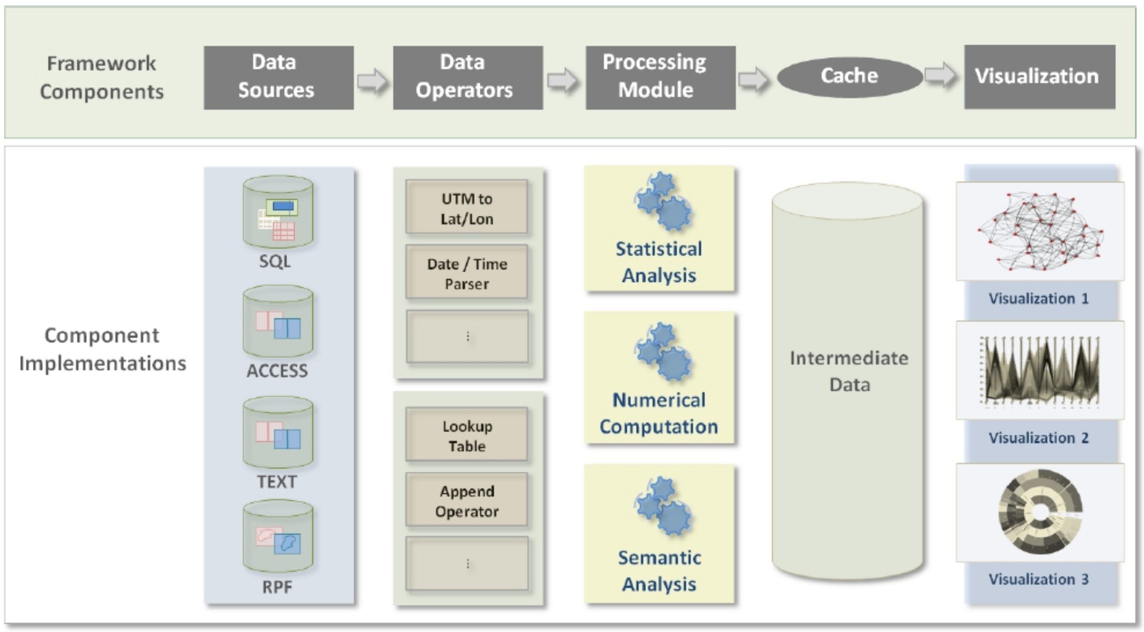


Fig. 5. The ProDV framework architecture defines *data sources*, *data operators*, *processing modules*, *cached intermediate data*, and *visualizations* to provide a simple visual abstraction of the visualization pipeline. The addition of *cached intermediate data* allows visualization applications to maintain interactive rates when working with large datasets.

files (like Microsoft Access) are common examples; however data sources can be defined from more custom data formats as well. For example, Raster Product Format (RPF) data, which provides standardized geo-rectified raster data, can also be used to provide tile data as records.

Data operators *Data operators* perform common data transformation or simple data aggregation tasks. Operators can represent simple data casting operations (e.g., parsing text as an integer), or be more complex such as a multiplexer or look-up-table that has an internal data structure.

Processing modules *Processing modules* encapsulate algorithms that generate reduced intermediate data structures from input data. Examples include simple numerical computations such as summation, statistical analysis, or multi-pass clustering algorithms. Many of our current processing modules perform and store computations at multiple resolutions, so that visualization controls can allow users to quickly change scale for different analysis tasks. Any of these objects can also define any number of user parameters.

Cached data While the general framework seems intuitive, one explicit design decision I will mention is the decision to not identify to users the specific details of intermediate data structures created by processing modules and stored as *cached data*. We have chosen to hide the details of visual mappings that exist between those data structures and the visualizations, since most users interviewed not familiar with the data structures that would be used (e.g., hash maps, trees, and heaps). Although it is difficult to evaluate this choice in a controlled experiment, I believe it simplified the user interface and enabled analysts with a broader range of skills to interact with the framework. User interface controls were provided for advanced users who did wish to manipulate visual mappings.

Visualizations A selection of interactive *visualizations* make up the final component of the framework. Each visualization defines a unique rendering method designed to be viewed in a standalone window. Several visualizations can be used to create coordinated multiple view layouts, as is common in visual analytics applications. Each visualization must define its own generic routine for building a render cache containing all the data needed to render the current view. Each visualization can contribute multiple view mappings that define how its render cache can be assembled from the data structures created by different processing modules. This architecture loosens the coupling between the processed data and the rendering algorithms, giving greater flexibility in the design of each.

5. Dataflow Interface

In order to simplify user configuration of the framework components, we developed a visual interface based on a visual programming environment we were familiar with, Eclipse. The visual interface was implemented using the Eclipse Graphical Editor Framework (GEF) and allows the user to connect and configure the data access and transformation components of the system, as shown in Figure 6.

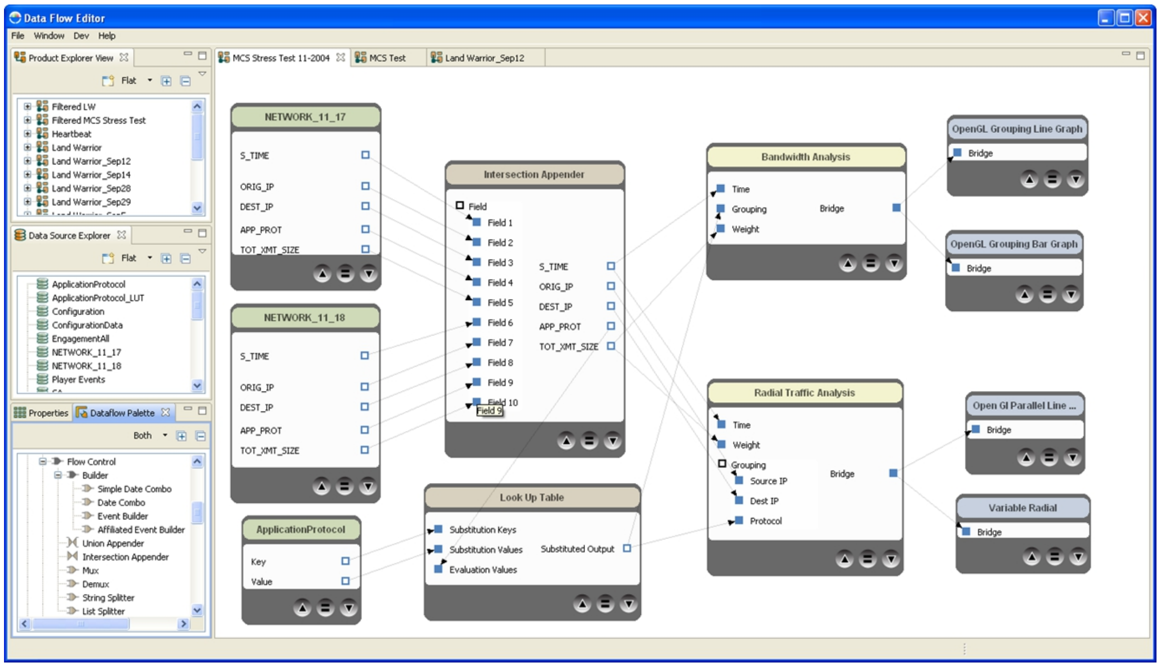


Fig. 6. ProDV Dataflow Editor interface. Example shows two similar data sources (green) appended and connected to a look-up-table using operators (brown). Two processing modules (yellow) provide cached data to four different visualizations (blue).

Windows containing user-defined data sources and framework components are shown on the left. Input and output field names and types are displayed and can be connected using common drag-and-drop interactions. Simple error messages are provided when the specified data model is invalid due to type matching errors or invalid or inaccessible data source configurations.

Once the user saves a connected group of components, referred to as a *product*, it can be launched in a separate window (similar to launching a new application in a software development environment). While the user can always adjust the data access or transformation configuration specified in the dataflow editor, any current cached data will be invalidated by this change since the state of the cached data and the product configuration are tightly coupled. When changes are made, the user will be prompted to close the current interactive visualization and any cached data will be cleared and recomputed using the new configuration.

6. Visualization Interface

Interactive visualizations created using the ProDV framework share a basic set of interactions. In order to promote consistency among interactions defined by our visualizations, we have defined a standard set of tools based on well-known information-seeking interactions: Select, Pan, Zoom, and Query. Each visualization can define hover, click, double-click, and drag mouse interactions using each tool, and only one tool can be used at a time. The Query tool provides mouse coordinates in screen-space, and any text returned by the visualization is displayed in a tooltip. Any other user parameters for a visualization can be adjusted in an auto-generated Properties tab, which shows user-adjustable properties for the current visualization or any selected object.

7. Implementation Summary

The implementation of the proposed method provided by ProDV meets the seven previously enumerated requirements for a visualization system designed to handle interaction with large numeric datasets.

Visual programming environment The visual interface shown in Figure 6 provides a visual programming environment that allows users to easily configure the components of the ProDV framework.

Multiple visualization methods A large collection of common visualization methods are provided by ProDV, including interactive line and bar charts, a radial axis chart, node-graph visualizations, and geospatial visualizations.

Coordinated views Visualizations in ProDV are coordinated using linked legend and timeline controls that enable global selection of data entities and time ranges.

Custom visualizations Custom visualizations can be easily integrated into the ProDV framework by simply defining the visualization's cache access mechanisms along with rendering code.

Unified visualization interface ProDV provides a simplified visualization interface using set of four tools used modally: select, pan, zoom, and info.

Data reduction Various types of processing modules implement data reduction using simple numeric aggregation or other computational algorithms.

Data caching Data structures produced by processing or visualization elements can be cached in memory and saved to disk to improve efficiency and enable collaboration.

End-user system ProDV has been designed to support domain experts without knowledge of programming concepts or data structures.

B. Case Studies

We have implemented a variety of visual analytics capabilities for domain experts using this framework. These capabilities included applications for network performance analysis, engagement analysis, and sensor performance analysis. In this section I will discuss the details of each of these analysis categories. For each section I will first discuss the analytic goals as indicated by domain experts, and then I will discuss our approach and the observed results. These cases show how the ProDV framework was easily adaptable to various analysis requirements specified by domain experts.

1. Network Analysis

Many of the tests conducted by these users involved networking equipment in some way. In most cases, networking metrics such as traffic volume per network protocol, Speed-of-Service, and Message Completion Rate needed to be analyzed for unexpected patterns over a large range of time. Since external software packages are available to compute and display common network performance metrics, we initially focused on providing the capability to correlate unexpected network anomalies and other user-specified features with these metrics. Later we implemented internal processing modules to compute MCR, SoS, and other custom network metrics.

In order to provide these capabilities, data sources from Access, SQL, text, and PCAP were used. We also used various operators for accessing packet data of various types (TCP, UDP, ARP, ICMP), parsing date/time and IP address information from text, and filters for bad or null values. The processing modules used here ranged from summing or averaging values over time to computing complex network metrics. The interactive visualizations include a line graph, a stacked bar graph, a parallel coordinate plot, and a radial chart based on the Radial Traffic Analyzer [34]. When available, we also included an interactive event timeline to depict both scheduled and unscheduled (via incident reports) events from the testing plan.

The network analysis capabilities provided using ProDV, shown in Figure 7, demonstrated effectiveness in two specific cases. First, after initial training using the ProDV

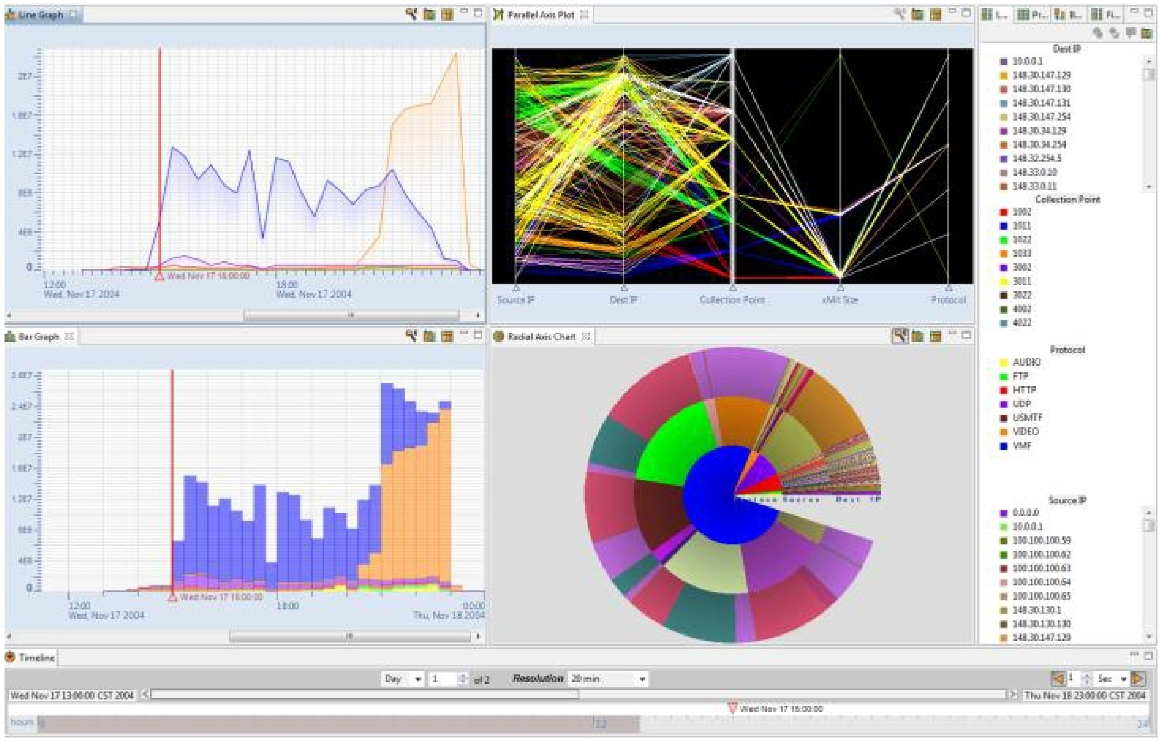
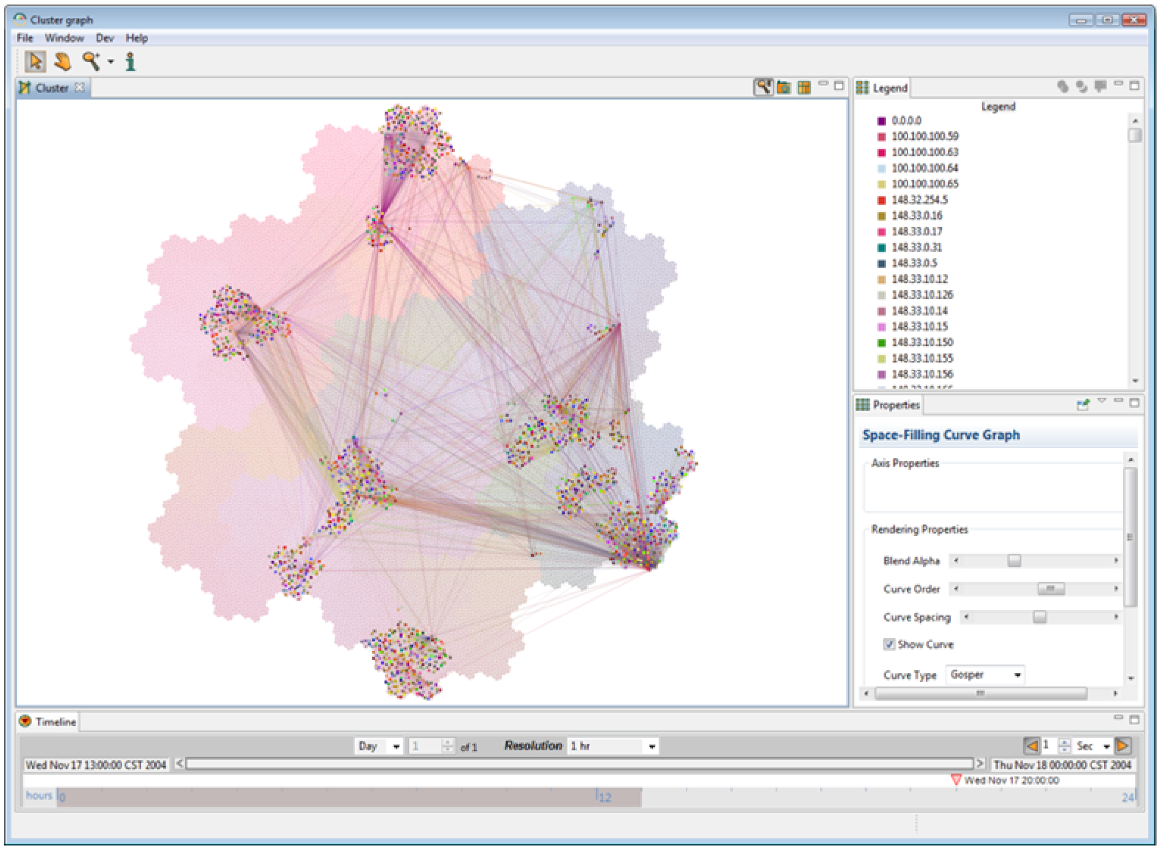


Fig. 7. An example interactive network analysis application similar to one built by domain experts using ProDV to validate data collected during a distributed simulation.

components to visualize data from baseline network events (running a common test script), on-site analysts were able to recognize expected features in the data. In particular, using the radial chart and line graph the analysts recognized the dominant senders and receivers of data within various segments of the network. After the training session, analysts were able to immediately recognize deviations from the expected network profile. In each case the discrepancies were due to lost or incorrectly processed data, and were able to be quickly fixed.

In another case, during the initial training with user data, three outliers (out of nearly 100 nodes) were immediately recognized, again using the Radial Traffic Analyzer, since they intermittently transmitted thousands of times more data than any other nodes on the network. It was quickly determined that these nodes were responsible for audio/video transmission and were able to be filtered out of many subsequent analysis activities.



Legend
- 🟪 0.0.0.0
- 🟥 100.100.100.59
- 🟥 100.100.100.63
- 🟦 100.100.100.64
- 🟨 100.100.100.65
- 🟥 148.32.254.5
- 🟫 148.33.0.16
- 🟥 148.33.0.17
- 🟩 148.33.0.31
- 🟦 148.33.0.5
- 🟧 148.33.10.12
- ⬜ 148.33.10.126
- 🟪 148.33.10.14
- 🟪 148.33.10.15
- 🟩 148.33.10.150
- 🟨 148.33.10.155
- 🟪 148.33.10.156

Space-Filling Curve Graph

Axis Properties

Rendering Properties

Blend Alpha

Curve Order

Curve Spacing

✓ Show Curve

Curve Type: Gosper

Day 1 of 1 Resolution 1 hr

1 Sec

Wed Nov 17 13:00:00 CST 2004

Thu Nov 18 00:00:00 CST 2004

Wed Nov 17 20:00:00

Fig. 8. Network cluster analysis application built using ProDV. Example shows clusters of network nodes with high traffic volume between them at the peak of a system stress test.

The example shown in Figure 8 allowed network analysts to examine the results of a cluster analysis of network behavior using the rapid graph layout technique described by Muelder and Ma [50]. This graph layout technique uses a previously published technique for clustering called “Fast Modularity,” and renders the graph nodes along one of several space-filling curves [12]. The cluster results highlight groups of network nodes that have significant traffic between them, and therefore allowed network analysts to verify expected results under varying degrees of network load and other varied conditions. The implementation also provided interactive features to scale node locations along the curve (allowing users to expand and collapse clusters along the curve) and animation. Because individual nodes were not guaranteed to be in the same location at any two timesteps, we introduced a feature to animate the motion of a user-selected set of nodes between timesteps. This allowed analysts to observe groups of nodes that were expected to cluster together over an extended period of time. In most cases, the clusters observed were relatively small (less than 50 nodes), and therefore analysts were able to validate the expected nodes in these groups without difficulty.

In all three cases we were able to successfully impart these analytic skills to domain experts, who continued use of ProDV in their analysis activities. One customer has since integrated ProDV into an internal suite of visualization tools used to validate simulation results.

2. Engagement Analysis

Several events provided engagement data from simulated kinetic engagements (kinetic actions are those involving direct force, such as firing weapons). In many cases a realtime casualty assessment (RTCA) system is used to determine the appropriate result of simulated engagements based on the attributes of the weapons and armor involved.

In this case we provided multi-dimensional visualization capabilities in the form of a parallel coordinate plot and a radial chart based on the Radial Traffic Analyzer [34]. Both visualizations depict the weapon type and the shooter and target of each engagement. The analytic goal in this case was to validate that the recorded engagements support the overall outcome as reported by the RTCA. However, certain interesting behaviors also became quite easy to detect using this analysis technique. For example, when simulation players were colored according to their affiliation (i.e., friendly units blue, hostile players red), friendly fire incidents were easily discerned. Also, analysts made use of the parallel coordinate plot to validate expected simulation configurations. For example, a one-to-one mapping was usually expected between a single engagement's ammo and weapon type. Variations, which were common, clearly indicated a misconfiguration. An example of this application is shown in Figure 9.

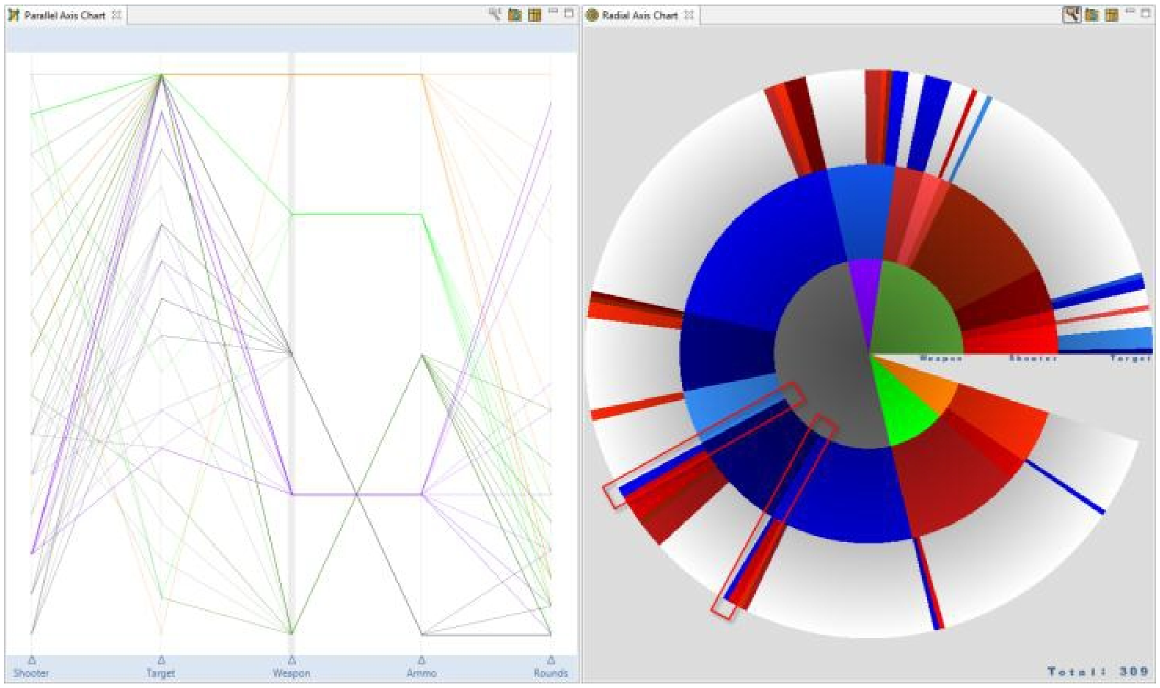


Fig. 9. Example of kinetic engagement analysis using a parallel coordinate plot and a radial chart. Two friendly-fire incidents are outlined in red in the radial axis chart on the right.

Based on requirements developed during interactions with domain experts, we also integrated custom spatial visualization of certain non-kinetic interactions such as message transmissions. This interactive geospatial visualization provided the capability to view the location of each transmission. The example in Figure 10 shows the purple player receiving two messages from other players who are in motion (red/white), and relaying the

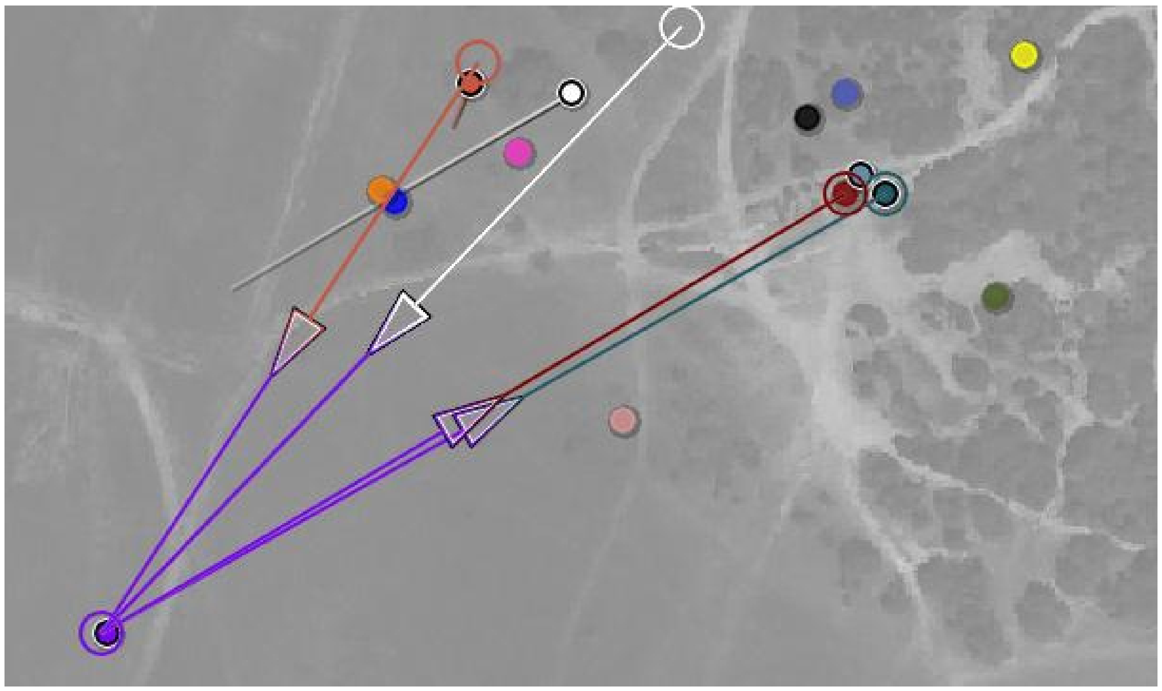


Fig. 10. Non-kinetic engagement analysis. Example shows successful completion of a message thread by stationary and moving players. The exact time of the individual message events can be accessed using a linked timeline.

message to two static players (maroon/blue). This capability is being used to evaluate the performance of mobile networking equipment for standardized message threads. Message threads consist of sequences of tactical messages related to a test activity. The messages should occur in proper order for some final action (e.g., call for fire) to be initiated. In this example the player successfully relays observation reports from scout players to positioned artillery players to complete the message thread, and the visualization allows the analyst to determine that the messages were sequenced correctly.

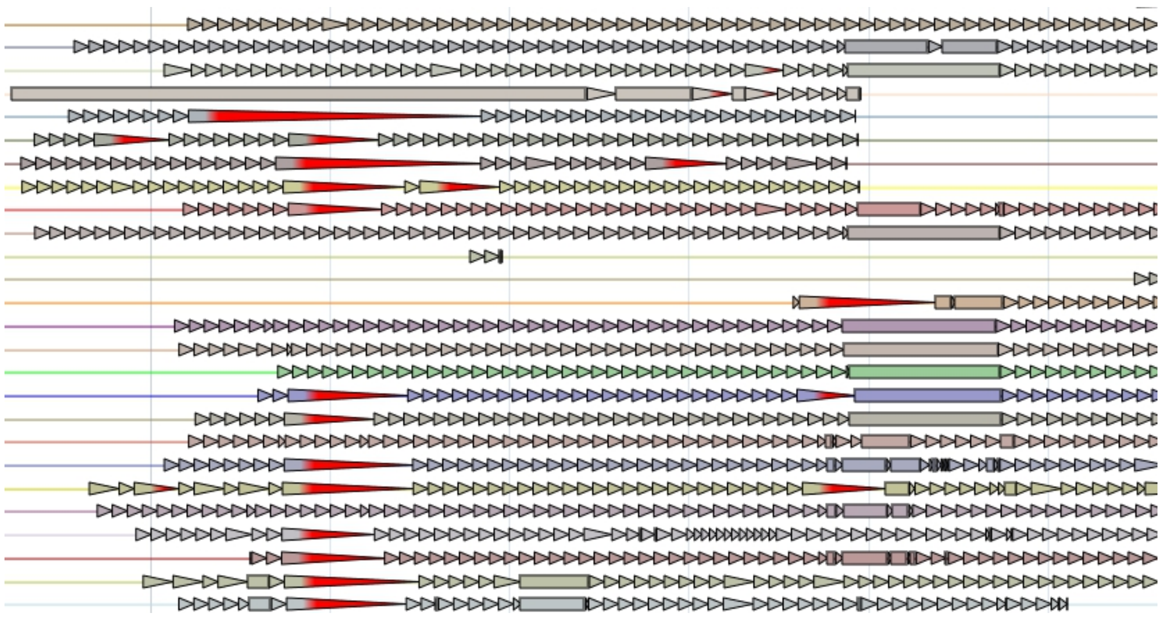


Fig. 11. Example of custom temporal pattern analysis visualization showing temporal activity patterns exhibited by a network of spatial sensors. Each row depicts the sensor reporting activity for a unique sensor, and each triangle represents a period of time between reports. Five minute threshold violations are highlighted in red, and triangles less than three pixels in width are merged to form quadrilaterals. A coordinated violation is shown by seven of the sensors in the beginning of the time period shown. The sensors used in this work increase activity when in motion, therefore in this example the quad areas represent spatial movement. A coordinated movement involving at least twelve sensors is shown in the middle of the period depicted, at the same time that five sensors stop reporting unexpectedly.

3. Sensor Performance Analysis

Performance analysis of distributed networks of spatial sensors over extended time periods was identified by our domain experts as a major concern. Therefore, we developed several visual analysis methods for both identifying anomalies in and conducting detailed analysis of sensor performance by leveraging existing capabilities and combining them with new, custom techniques.

a. Temporal Anomalies and Pattern Recognition

First, we developed a custom visualization to support analysis of temporal activity patterns within a network of spatial sensors, shown in Figure 11. This visual analysis technique was based on user requirements specified during interviews and a software training session. It allowed data collectors to quickly discern unexpected sensor failures from coordinated maintenance activities, and explore and analyze expected temporal patterns generated by coordinated movements within the sensor network. This technique will be discussed in more detail in Chapter V.

b. Data Collection Analysis

Another interaction with domain experts led to the development of a custom processing module that computes a defined heuristic in order to evaluate the performance of a sensor data collection system. The data collection system was designed to capture data using three redundant storage systems, and analysts needed to know when redundancy was being sufficiently achieved. The results were visualized in a simple state plot and showed when each data collection system was operating with full, partial, or no redundancy. This capability was developed and delivered rapidly, and quickly led to the discovery of specific bottlenecks in data throughput that caused the results shown in Figure 12.

4. Geospatial Analysis

We worked with domain experts on a variety of geospatial analysis problems dealing with sensor performance. In most cases we began by focusing on anomaly detection. The most prevalent anomalies we encountered in geospatial datasets were incorrect or out-of-sequence locations being reported by spatial sensors in motion. For example, players observed conducting a convoy route intermittently reported an anomalous location (often

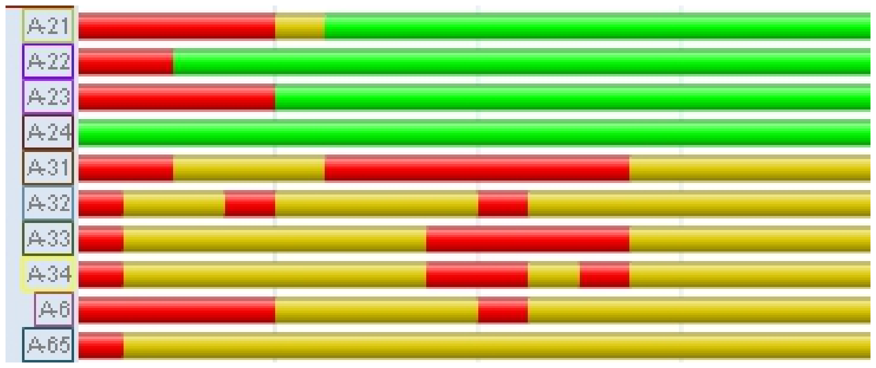


Fig. 12. Data collection system analysis results shown in an interactive state plot. Color (green, yellow, red) indicates state of data collection system redundancy (full, partial, none).

some initial location) along the route. To facilitate a detailed analysis of these anomalies, we were able to rapidly assemble a new interactive visualization by complementing the interactive geospatial visualization with existing capabilities that had been previously introduced for network analysis.

Using the animated geospatial visualization, shown in Figure 13, we were able to identify a temporal pattern of sensor activity correlated with spatial reporting anomalies.This temporal pattern helped analysts identify other anomalies using the temporal analysis plot. It was observed that in most cases the anomalies occurred in areas with heavy foliage. Therefore this analysis led to the conclusion that loss of wireless network connectivity caused this type of sensor to generate a series of anomalous reports which exhibited a visually recognizable failure pattern. This pattern was subsequently used to pinpoint environmental conditions (i.e., foliage density) that contributed to these failures.

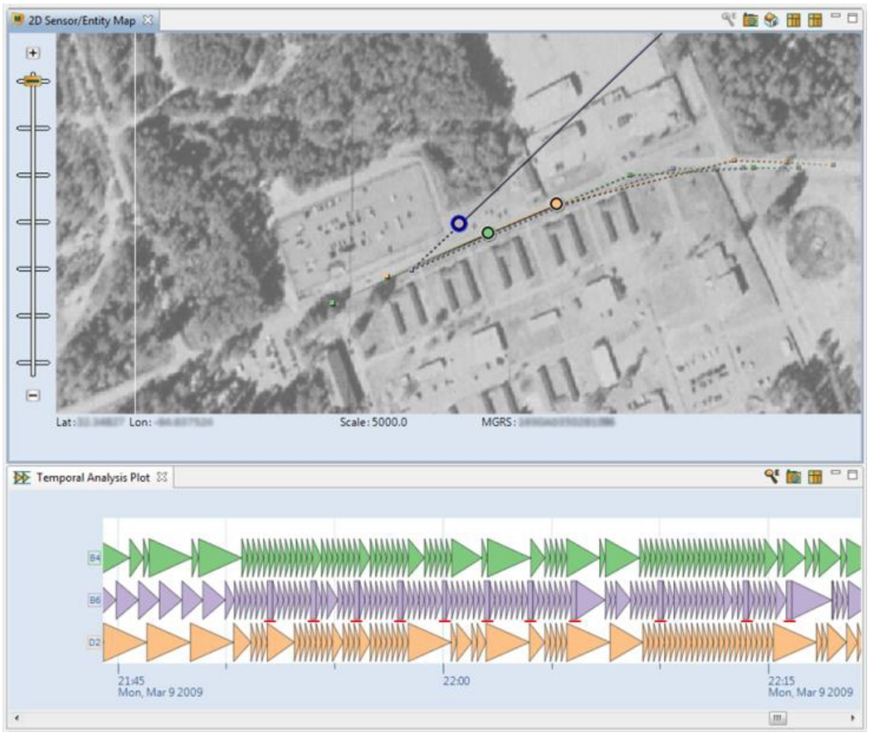


Fig. 13. Spatial sensor performance analysis application. The highlighted temporal pattern (red lines under the second row) was recognized as an indicator of a particular sensor failure.

5. COP Currency

We have also implemented a custom visual analysis capability defined by our expert users. They have expressed particular interest in comparison of the Common Operating Picture (COP) viewed by each player in a simulation exercise. The COP defines the player's current perception of the location of all other players in the simulation. From our perspective, the COP was viewed as the state of each player's data containing the most recently received locations for all other players, often called Situational Awareness (SA) data. To this end, we defined a metric, COP currency (CC), such that for player X:

$$CC_x = \frac{\Sigma_1^n \left\{ \begin{array}{c} 1 \text{ if } \left|p_i - p_{perceived\ by\ X}\right| < d \\ 0 \text{ otherwise} \end{array} \right\}}{n} \tag{3.1}$$

where d is a user-specified threshold distance and n is the number of other players. Simply put, a player's CC value is high if all of his current SA data is within tolerance. Otherwise, the CC value will fall off proportional to the number of players. Current work is aimed at providing additional mechanisms for computing COP currency according to pre-defined player hierarchies. That is, analysts wish to constrain the COP currency computation for unit X to n specific other players (e.g., a squad leader's COP currency value is only affected by data from players in his squad). It is straightforward to incorporate new currency metrics into the system, since each would require only a new processing module, and no change to the visualization or other modules.

The goal of this analysis is to identify potential correlations between increased network latency and degraded COP currency. Since wireless players experience varying network latency under varying terrain conditions as well, we display a graph of the COP currency metric along with a geospatial visualization that allows the user to display the COP of any selected player. Only one COP can be displayed at a time, and the display shows the perceived positions and actual positions. The example in Figure 14 shows a particularly low point in CC for a particular player. The details of this technique will be discussed in Chapter V.

C. Evaluation

The success of visual analytics methods has been evaluated by researchers in this community in a variety of ways. During the course of this work, I have evaluated the effectiveness of these methods using three different evaluation techniques.

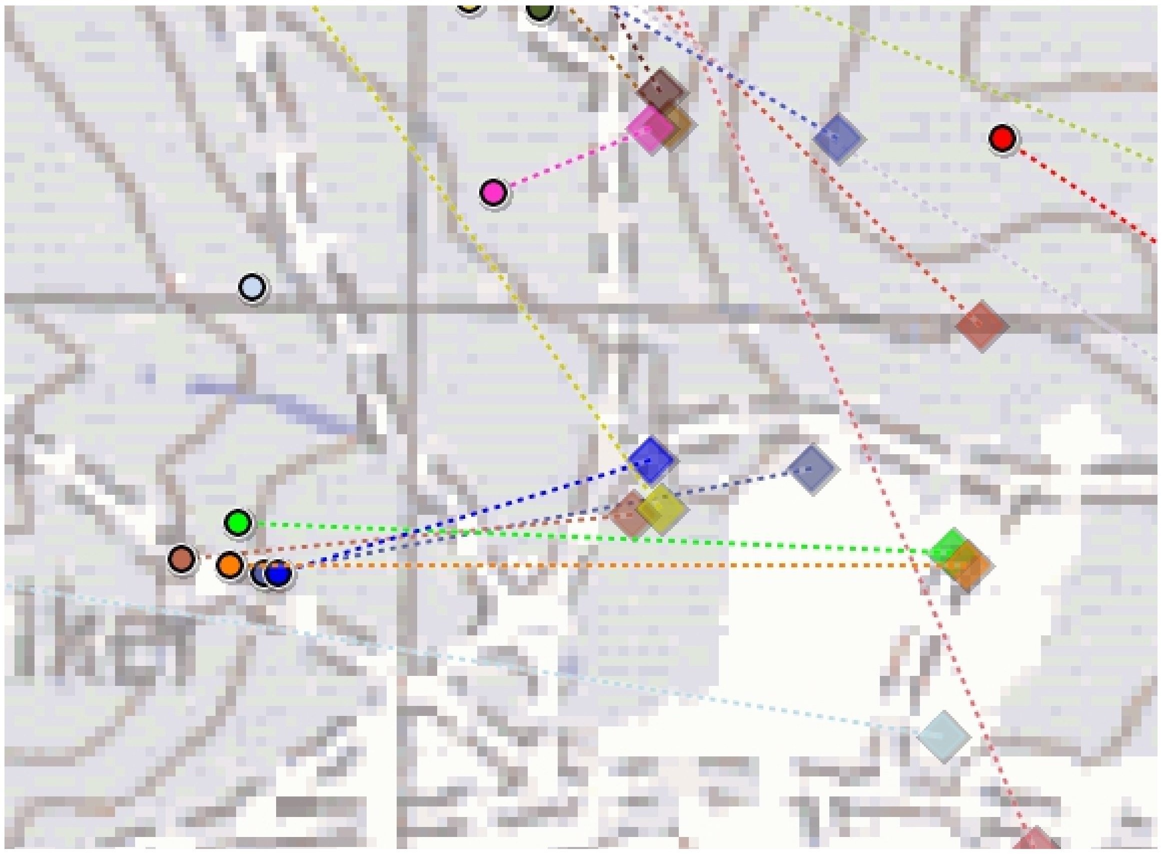


Fig. 14. COP currency visualization. Circles indicate current player locations, diamonds indicate locations perceived by selected player's COP.

First, I have quantitatively evaluated the performance in terms of speed and accuracy of volunteer participants using specific visualization techniques to analyze data in a controlled environment. While these evaluation techniques offer some benefit for judging or comparing individual visualization techniques, they do not provide a basis for evaluating the entire system. In particular, the pool of participants is limited to non-experts, and the techniques cannot be evaluated in terms of true workflow.

As an alternative, I introduced simple evaluative techniques within the training program designed to introduce the customer domain experts to the ProDV software. While there are challenges to this approach, it does provide data from customers and domain experts in a more operational setting, which is arguably more relevant when evaluating a visual analysis technique.

Finally, I also will present expert opinion extracted from both training sessions and other domain expert interactions to demonstrate a high level of understanding and satisfaction with the ProDV framework concept.

1. Individual Visualization Evaluation

We conducted a series of task-based user analysis studies aimed at evaluating the effectiveness of a custom visualization of temporal sensor patterns (Figure 11) using small groups of graduate students. The tasks for the user evaluation study were based on detection of the temporal and spatial anomalies in the context of sensor performance analysis. For example, users were asked to first temporally locate sensor failures within coordinated activities, and then determine the location of the sensor prior to the failure.

While the full details of the results of this evaluation method are discussed fully in a later section, our study indicated decreased time and increased accuracy for both temporal and spatio-temporal analysis tasks. However, since these results came from non-expert users, I feel that this evaluation mechanism is not sufficient to evaluate the effectiveness of the framework as a whole.

2. Expert Training Evaluation

In order to evaluate how well expert data analysts were able to comprehend and interact with our visualization environment, I designed simplified task-based evaluations that were added into software training sessions. The training sessions were already being used to incrementally introduce new users to each of the analysis modes discussed. I added three similar analysis tasks to the end of each session, and implemented an interface within our software to record the time spent on each task. The task responses were recorded manually by each user on a form. Since the goal was to evaluate how well expert users were able to successfully assimilate various analytic skills, each session focused on the different analytic skills required for each task.

The results of the first experimental training session were mixed. First, although nearly all users completed the tasks at the end of each session, there were significant problems using the embedded timing interface and therefore only two timing records were complete from a total of twelve participants. This experience provided a valuable lesson. In attempting to evaluate user performance in the context of a software training session, it was necessary to avoid distraction from our primary purpose of providing a step-by-step introduction to the basics of the software. The incomplete timing records indicated that our evaluation mechanism was most likely a large distraction in its current form. Based on the problems observed using the initial embedded interface, I determined that requiring users to manipulate a simple interface for timing while writing responses on an evaluation form was too distracting. Because this task was not an integral part of the training experience, the trainees would often forget or ignore the action during the session. Therefore, I redesigned the embedded interface to provide the task descriptions and prompt the user for a response (both the response and the time are recorded automatically). This minimized distraction while automating the recording process for our next training session.

I recognize the benefits of this method of evaluation since it allowed me to collect data on multiple expert users simultaneously, which I believe is superior to conducting additional controlled experiments using graduate students.

3. Expert Opinion

I also collected expert opinions using query forms during several training sessions. These opinions were given after training sessions that covered all of the previously mentioned analysis techniques over approximately a six-hour period. The experts indicated that:

- the analysis capabilities provided were “excellent” and “very mission related.” This reinforced the experts’ early involvement in the design and development process of the ProDV framework.
- overall, the software was “very user friendly—mouse [interactions are] familiar.” This reinforced the interaction constraints (limited toolbar) we adhered to in our design.
- some of the visual elements of the data flow interface were difficult to read. This issue was addressed by allowing user modification of line thickness and other attributes.
- they would have liked more time to explore the datasets provided. This reinforced our use of sanitized real-world datasets provided by the customer, in that these datasets were sufficient to teach basic concepts, but were not too involved to distract or confuse students from the goal of the lesson.

D. Collaborative Analysis

The architecture of our framework has allowed us to implement several useful collaborative analysis features as well. These are divided into annotations and content sharing features. Annotations provide additional semantic content on any object defined in the framework. Content sharing features allow users to share any of the framework objects via file export or directly via local network, including annotations. I will discuss how together shared content and annotations enabled basic collaborative analysis activities.

1. Annotations

We have made use of annotations in several forms to enable collaborative analysis activities within the ProDV framework. First, we allow users to add comments to any framework components. This allows users to describe conditions for certain data sources, make comments about existing products, or even describe in detail the justification for a specific operator's configuration.

In addition to semantic annotations, we also allow the user to save and annotate a specific view within the interaction space using a *bookmark*. A bookmark in the ProDV framework encapsulates all current view parameters including zoom, color assignment, filter state, view layout, timeline attributes, and object selections.

2. Content Sharing

By sharing a product and its cached data, another user can easily run an interactive visualization application created and processed by another user. This allows multiple analysts to collaborate on the analysis of a large volume of data by creating and sharing analysis applications for smaller subsets or contexts of data.

In addition, users can share the annotation mechanisms discussed in the previous section. Sharing of bookmarks and comments allows users to save and share with remote peers additional semantic information about certain data features. Together, these features allow an analyst to discover an interesting feature using a combination of interactive visualizations, bookmark and comment on the feature, and share the interactive visualization with another analyst who would be able to contribute to the understanding of the data.

3. Evaluation

Evaluation of these techniques poses a problem for us. Specifically, these features involve network protocols, which the customer requires meet a stringent certification process. This limits our ability to automatically record and report user data (as could be done for the standalone analysis), and slows down the ability to gather feedback from experts. An area of ongoing work is determining an effective method to gather such collaborative data under the certification constraints imposed.

E. Ongoing Work

In this chapter, I have presented an interactive framework that effectively enables domain experts to customize a variety of visual analysis capabilities in order to perform analytic tasks in several different domains. There is continuing work with domain experts to evaluate any long-term benefits of our approach. It is expected that ProDV will continue to be used for analysis tasks by these users, as well as to support other current research efforts in human social-cultural behavior (HSCB) modeling and visual analysis applications for the electro-magnetic spectrum (EMS). To date, ProDV has provided the basis for over $2 million in U.S. government funding for projects focused on research and development of custom interactive visualization software.

CHAPTER IV

ASYNCHRONOUS VIEW-DEPENDENT STRATEGY FOR SPATIAL DATA

In this chapter, I will discuss work aimed at improving interactive visualization systems when working with high-resolution terrain imagery data that must be retrieved at runtime from a remote database. The technique I have proposed is aimed at enabling visualization systems designed to operate in such an environment to efficiently prioritize and retrieve the minimal set of data required to render a visual representation of a spatial environment, while maintaining interactive rates that allow the application to be useful to complex analysis tasks.

Many challenging analysis tasks in science and technology require massive amounts of high-resolution data collected from various sensors. Examples of analysis challenges requiring such high-resolution terrain topology and imagery include interactive exploration of near-earth RF propagation models, coordination of disaster recovery efforts, military operation planning, and excavatory operations. The introduction of commercially-available sensors that can provide such high-resolution data for a small area, as well as increased commercial satellite imagery and terrain resolution for certain areas, makes this work particularly relevant. In order to be useful to analysts performing these interactive analysis tasks, visualization software applications using this data must retrieve the data efficiently from a remote database. This work is focused on using high-resolution terrain data from the National Geospatial-Intelligence Agency (NGA) to support detailed interactive analysis of various geospatial datasets. I will discuss use of the highest resolution data openly available to government agencies in the form of Digital Terrain Elevation Data (DTED) and Controlled Image Base (CIB) satellite imagery. By defining a strategy to efficiently pre-allocate and index areas of video memory, a visualization system can be optimized to retrieve texture and elevation data at full resolution in order to best support interactive visual analysis. The proposed method allows a visualization application to efficiently allocate video memory, retrieve the highest-priority data (defined by the interactive view), and load the data asynchronously, while minimizing the impact on rendering performance.

A. Problem Description

The motivating task for this work is the efficient implementation of an interactive three-dimensional spatial data visualization environment that utilizes a large remote terrain database that provides both terrain elevation and imagery at multiple resolutions. The terrain database cannot be stored locally and therefore retrieving data within the region of interest will incur an unknown latency. In the case of high-resolution imagery, texture memory resources are likely insufficient to store all the data referenced by the current view. This method provides a solution that will load as much terrain data at full resolution as possible, in a view-prioritized fashion, instead of decreasing the level of terrain detail in the scene. This method has been used to complement additional visualization layers of spatial data for visual analysis, such as recorded GPS data for multiple ground-based spatial sensors visualized as three-dimensional trails. Such an interactive visualization is useful in analysis tasks such as detailed inspection of the effects of terrain and vegetation on the performance of mobile ad-hoc networks. During such analysis, the interactive view must be able to rapidly transition from inspecting small-scale features such as vegetation to large-scale features of terrain topology and from one location to another. In this chapter I will demonstrate how this method allows analysts to interactively explore regions of interest at full resolution within a massive remote dataset. This work can be directly applied to high-resolution geospatial datasets that are of particular interest to various parts of the scientific communities, including seismic and electromagnetic spectrum data, to name a few. The timely visual analysis of these types of data is a tool of growing importance for analysts in many domains [13], including the military, disaster recovery, and petroleum industries. The techniques discussed here can also be adapted and applied to other visualization domains, such as medical and genomic visualization.

B. Proposed Method

The traditional approach to developing an interactive visualization system using a large remote database is to pre-load data at various pre-processed levels-of-detail so that appropriate LOD data structures can be sampled and a simplified representation of the scene can be rendered. The total geometry count in the scene may vary as the interactive visualization algorithm adjusts the level-of-detail, affecting the rendering performance of the system. There are several disadvantages associated with this approach. First, as the level-of-detail changes, rendering performance will vary as the user navigates the environment, although there are algorithms designed to minimize this effect by keeping overall amount of scene geometry stable [46, 15]. Also, preprocessing routines for generating multiple level-of-detail geometry and texture data can be complex and computationally expensive. Finally, the simplification algorithms used to simplify this data will by definition distort the original data. The drawbacks to these simplification methods are acceptable when rendering objects for visual effect (such as for interactive gaming or other entertainment purposes) and balanced by increased rendering performance, but may cause unwanted inaccuracy when rendering data for detailed scientific analysis.

This approach addresses these challenges by leveraging the visualization pipeline to prioritize data requests in a view-dependent manner, using custom vertex and fragment programs to sample data indices aligned in pre-allocated texture memory. This enables the visualization system to maintain a static amount of geometry in the scene (pre-allocated in a vertex array) and static texture memory allocation to achieve stable rendering performance.

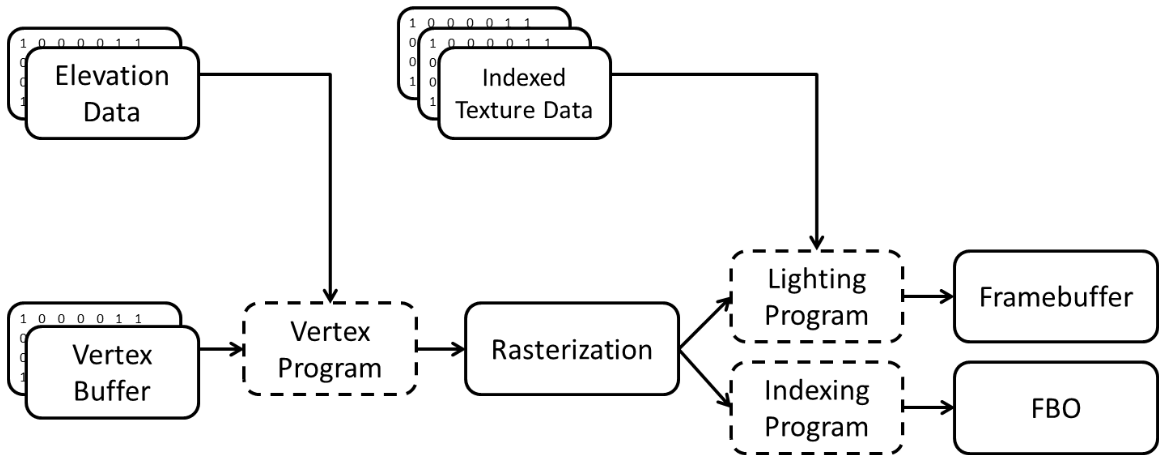


Fig. 15. The proposed method leverages the architecture of the graphics hardware pipeline to efficiently view-prioritize data requests by aligning index data into texture memory and rendering data indices into an off-screen *frame buffer object* (FBO) by simply swapping out the fragment program but not swapping texture memory. The programmable components of the pipeline are shown with dashed outlines. The framebuffer output shown in the figure is what is shown on the user's monitor.

A graphic representation of this shown in Figure 15. The proposed indexing method also enables prioritization of the data indices referenced by the current view using the hardware-accelerated rasterization pipeline, thereby eliminating the need to compute view priority on the CPU. In order to maximize performance on current graphics hardware, this method minimizes the total number of texture elements used concurrently (for which there is a finite limit on current hardware implementations) and the total number of changes in texture bindings (which reduce rendering performance).

Because the latency of data retrieval from a massive remote dataset is unknown, data should be retrieved asynchronously so that the rendering thread is never blocked and therefore interactive rendering performance is not impacted. Because the priority scores are output by the graphics pipeline, this technique is flexible enough to handle rapid changes in view priority at any time, which is essential for supporting exploratory analysis of spatial datasets. The view-dependent function used to compute priority scores can be defined based on pixel count, screen location, and depth.

Because the amount of video memory available varies with every platform, dependent upon the capabilities of the local graphics hardware, this method is adaptable to make efficient use of that memory. There is an important distinction between *virtual* video memory and *addressable* video memory. For the purpose of this work, *virtual* video memory is defined as the amount of memory that can be allocated by our OpenGL context at runtime, while *addressable* video memory is defined as the amount of memory that can be concurrently bound in OpenGL (and therefore referenced by an active rendering context). The amount of addressable video memory will always be less than the amount of virtual memory, and therefore all allocated memory objects are not always available without paging from main memory. Furthermore any change in use of addressable memory requires rebinding of texture memory, which introduces additional delay to the rendering thread. This technique optimizes use of both memory types while minimizing changes in addressable video memory to maximize rendering performance when managing large amounts of high-resolution data. View priority scores are used to determine which virtual memory references will be bound at any given time.

C. Implementation

In this section I will discuss the interactive implementation of the proposed asynchronous, view-prioritized data retrieval technique. I have developed this implementation using a Java-based windowing system (SWT), JOGL OpenGL bindings, and GLSL vertex and fragment programs. This implementation also makes use of custom spatial database services based on the OpenMap function libraries [66].

1. RPF/DTED Data Structures

Raster Product Format (RPF) and Digital Terrain Elevation Data (DTED) are standardized data formats defined by the US Department of Defense [69, 70]. RPF and DTED datasets at various unclassified resolutions are available to all U.S. government organizations through the National Geospatial-Intelligence Agency (NGA). This implementation uses OpenMap, an open-source Java-based project available from BBN [66], to access and query these datasets. For this work I have provided local and remote custom database services using the capabilities provided by OpenMap.

a. Raster Product Format

Raster Product Format (RPF) datasets are normally distributed via CD/DVD as collections of RPF products at multiple resolutions for a given area. Most RPF datasets include Compressed-Arc Digital Raster Graphics (CADRG) cartographic maps, including topographic line maps (TLMs). Compressed Image Base (CIB) datasets include monochromatic geo-rectified satellite imagery at 5- or 1-meter resolutions. Each dataset contains a table of contents file that describes the map properties and coverage areas. All RPF data is composed of 256x256 pixel tiles of 8-bit indexed color, each map set providing its own color palette (many of these consist of 16 colors or less).

b. Digital Terrain Elevation Data

Digital Terrain Elevation Data (DTED) datasets are composed of individual frames of elevation data organized in a regular grid covering a full degree of latitude and longitude. DTED data is classified by level, where a lower level contains a lower resolution (fewer elevation readings) of data. DTED level 1 is commonly referred to as DTED1, and so on. Regardless of DTED level, each frame will always contain a full degree of latitude and longitude. For the purpose of this work, we mainly consider use of DTED1 and DTED2 data, which provides elevation postings at 3 arc seconds (approximately 100 meter) and 1 arc second (approximately 30 meter) intervals, respectively. Therefore, a DTED2 frame (3600×3600) is 9 times larger than a DTED1 frame (1200×1200). DTED elevations are stored as 16-bit signed integer values, which represent the elevation in meters relative to mean sea level.

c. Retrieval Latency

For the purposes of performance evaluation, I have implemented both local and remote versions of our OpenMap based query engine. While in most settings the remote implementation is most practical due to the space requirements to store large collections of RPF and DTED data, I will evaluate this implementation using both versions to show how the system performs with varying latency of data retrieval. For general comparison, the local database is usually on the scale of a few GB, while the remote database is currently housed on a 2 TB dedicated RAID array and contains several hundred GB of elevation and imagery data. The remote data service utilizes compression techniques to minimize the cost of transmitting both RPF and DTED data blocks.

2. Scene Layout

The three-dimensional scene consists of multiple regular grid structures representing rectangular blocks of terrain data. I define a static triangle loop iteration through a vertex array of generic vertices, and a vertex program computes the final location of each vertex based on the vertex id and the desired terrain resolution. The vertex array defines 1200×1200 vertices, according to the size of a DTED1 frame. DTED2 data, which provides three-fold resolution of DTED1 data in each dimension, can be rendered by iterating the vertex array nine times while adjusting latitude and longitude offsets. Other spatial data may be drawn in addition to the terrain, and elevation values can be easily be sampled from the texture data during these rendering operations.

3. Texture Memory Objects

Video memory is optimized for storage of rectangular blocks of texture (i.e., color) information. In this section I will discuss a strategy for constructing indexable memory blocks in video memory for storing both elevation and raster data.

a. Elevation Data

DTED elevation data can easily be stored in texture blocks with a few simple considerations. First, the 16-bit precision of the data must be preserved. Therefore, I use a 16-bit monochrome texture array. Some older hardware implementations may not provide this format, and in this case another scheme such as a two-channel 8-bit format or a 5-5-6 RGB format can be used. Bit-shifting operations can also be used to decompose and reconstruct our 16-bit values. This implementation did use a 16-bit mono texture format, however this impacted the choice of indexing techniques.

The terrain visualization implementation renders a full degree of latitude and longitude centered around the current viewpoint, and therefore four frames of DTED data are required in memory at any time, as shown in Figure 16. By allocating a large 16-bit monochrome texture, each of the four required frames can be stored in a separate quadrant of the texture. I also maintain a 4×4 matrix that also defines (in row-major form) the lat/lon bounds and size (1200/3600) of each frame, which is supplied as a matrix input to the vertex program. The matrix is updated as the viewpoint location changes; new frames are swapped in to replace unneeded frames.

Newer graphics hardware implementations do provide large texture blocks up to 8096×8096. Allocating a texture of this size allows us to be able to load either a DTED1

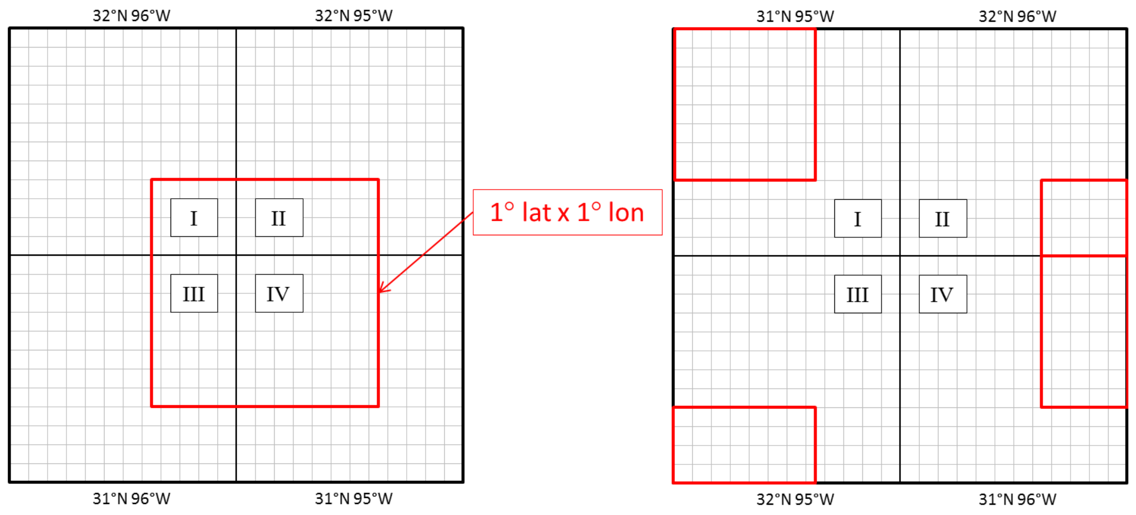


Fig. 16. DTED memory structure. I allocate sufficient memory to store four full frames of DTED elevation data. If sufficient size texture elements are available, DTED2 resolution can be stored without use of multiple textures. A 4x4 matrix input is used to define the configuration data required to sample the individual quadrants. The figure on the left shows the four frames of elevation data neatly juxtaposed. In practice the frames will be placed in quadrants based on the order required, resulting in cases more like the figure on the right.

or a DTED2 frame into each quadrant, and using the frame matrix frames of different resolutions can be stored in different quadrants. Older hardware implementations that only provide up to 4096x4096 textures should restrict the implementation to using only DTED1 data or use multiple texture objects to store the elevation data. Since current hardware implementations provide a limited number of concurrently active texture bindings, it is preferred to use only one texture object for elevation data when possible. Of course there is a trade-off here since some allocated texture memory certainly goes unused on hardware platforms that do not provide non-power-of-two texture sizes. In this implementation I do make use of such texture sizes, available on most contemporary high-end graphics hardware, and avoid unused allocated memory by using a texture size of 7200×7200 to store four frames of DTED2 data.

b. Raster Data

Since RPF data is color index data provided in power-of-two sized blocks, this data is very conducive to storage in texture memory. However, in order to reduce the number of active textures to be bound when rendering, I have combined the smaller 256×256 tiles into larger texture blocks, which I refer to as *compound tiles* (see Figure 17). The total size of compound tiles can be adjusted, but are usually 2048 or 4096 in practice, allowing 64 or 256 tiles to be compounded, respectively.

A compound tile exists in three distinct states. First, it is realized as metadata only, meaning that no texture memory is currently assigned for that area. The compound tile metadata defines the latitude and longitude bounds of the block and is assigned a unique index. Second, the compound tile can be assigned a currently allocated compound tile, which is not yet referencable by the rendering context. And finally, an allocated compound tile containing high view-priority data can be bound (addressable) for rendering. The life cycle of the compound tiles is determined by tile coverage results from spatial OpenMap queries. As these results are processed, each new tile is assigned a unique index (tile indexes and compound tile indexes may overlap). If a new tile is encountered that is not contained by an existing compound tile, the metadata for that compound tile is defined. The compound tile will be assigned texture memory (i.e., allocated) and/or bound when view prioritization dictates. As each frame is rendered, compound tile assignments and bindings will be determined by view-priority scores, and by the amount of available virtual and addressable video memory available.

4. Indexing Techniques

In order to determine the view priority of the indexed terrain data interactively, I align and store tile and compound tile indices within the allocated video memory for both elevation

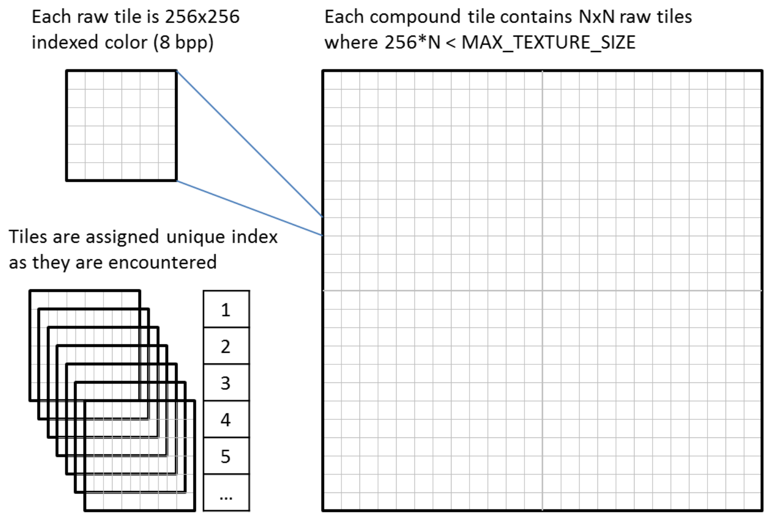

Each raw tile is 256x256
indexed color (8 bpp)
Each compound tile contains NxN raw tiles
where 256*N < MAX_TEXTURE_SIZE
Tiles are assigned unique index
as they are encountered
1
2
3
4
5
...

Fig. 17. RPF memory structure. Compound tiles are composed of multiple 256×256 tile elements. Each tile and compound tile is assigned a unique index.

and imagery data. By embedding the indices of image tiles and their compound parents into the video memory, the indices can be sampled during rasterization and subsequently prioritized based screen coverage, location, and view depth.

I implemented two distinct techniques for storing tile indexes alongside the raster and elevation data structures. Either distinct index textures or mipmap memory can be used to store the indices. In either case, it is important to consider the size of the indexable space. For example, 24-bit indexing can provide 2^{24} (approximately 16 million) distinct indexes, while 32-bit indexing can provide 2^{32} (approximately 4.2 billion) distinct indexes. For 1-meter imagery CIB datasets, 24-bit indexing is both required and sufficient.

a. Mipmap Indexing

The desired approach to embedding index data into the data structures does not require a separate texture, since there is a finite limit on the number of textures that can be concurrently bound (and therefore accessible from the rendering context with no state change). This limit is usually 32 for most current hardware (although when using large textures all 32 still may not be able to be successfully bound). Therefore I map our indices into the mipmap memory areas of the texture, which are bound along with the main texture memory [74]. Since the base texture as discussed is an 8-bit per pixel format (for indexed color), multiple mipmap levels are required to store a 24-bit or 32-bit index, as shown in Figure 18. In this case I have ensured that highest mipmap level used (smallest size in texels) still provides an appropriate resolution to represent the index coverage. In this implementation, I have used 4096×4096 compound texture tiles, and therefore 16×16 or 256 image tiles can be loaded into each compound tile. By using the first 3 mipmap layers, we can guarantee adequate resolution to store index values because unique indexes can be specified for each $2^n \times 2^n$ or 8×8 block of pixel data where n is the maximum mipmap level used ($n = 3$ in this case, and distinct tiles are 256×256). Custom vertex and fragment programs are used to sample the values at each mipmap level and reconstruct the 24-bit index value. A careful software implementation can still make use of some mipmap levels in between the index levels and the base level to increase texture sampling performance.

5. Loading Indexes

I have identified two distinct methods for loading data indices into texture memory, whether using a separate index texture or using mipmap indexing. The direct method uses the standard OpenGL texture loading functions [*glTexSubImage2D()*] to load the indices [74]. This requires an array to be allocated in main memory and populated with appropriate index values prior to loading, which in turn requires interpolation functions to be executed on the CPU to determine coverage bounds for each index individually. Alternatively, we can attach a *frame buffer object* (FBO) to our texture memory, and render each index directly into the texture by defining the current color as the index value and simply rendering a quadrilateral primitive corresponding to the tile or compound tile bounds using standard direct-mode OpenGL rendering commands [*glVertex2D()*]. While this approach is simpler

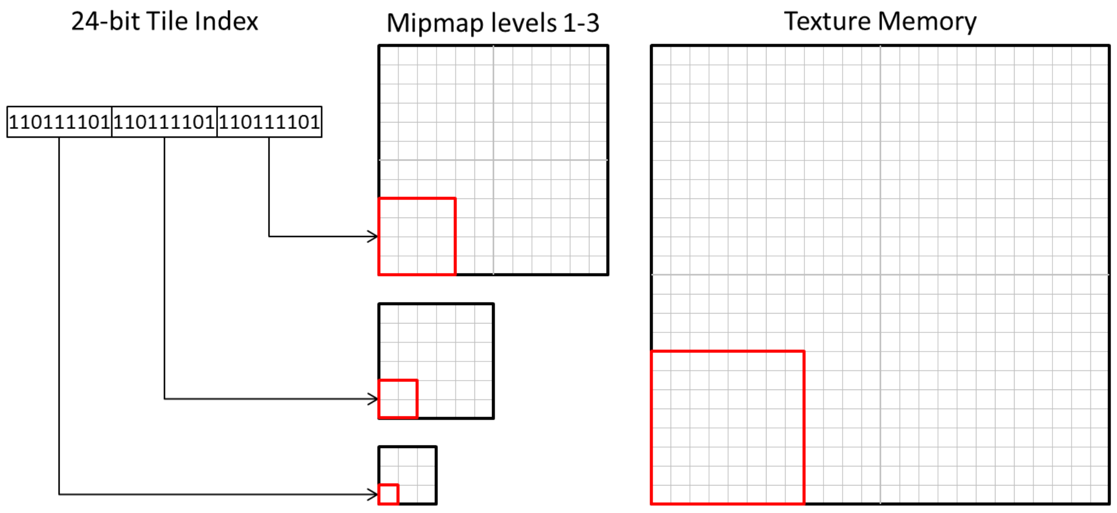


Fig. 18. A 24-bit index is split into three 8-bit components in order to be stored in multiple mipmap levels. A custom fragment shader can reconstruct this index by sampling each level. The highest mipmap level (smallest size in texels) must still provide adequate resolution to avoid aliasing when sampling indices.

to implement and does not require allocating and pre-populating redundant index values in main memory, current graphics hardware implementations vary on support for attaching FBOs to different texture memory formats. As a result the index rendering method may not be available, depending on the hardware platform support for FBOs using the index texture format. If FBOs are available for the desired texture format, the implementation must attach a unique FBO to each mipmap level of the texture, and render each byte of the index separately.

In this implementation, I have used a non-standard texture format (mono 16bpp) to store elevation data. None of the current graphics hardware tested during this work allowed attaching a FBO to a texture of this format. Therefore, I chose to use a separate texture object to store compound tile indices aligned with elevation data (see Figure 19). I use a standard 32bpp RGBA texture format that could be rendered into directly by attaching an FBO. Because of the finite limit on the number of bound texture objects, in most cases it is

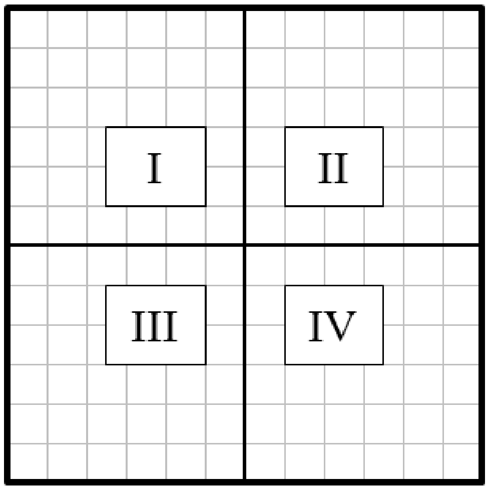


Fig. 19. This separate texture memory object is similar in structure to the elevation texture object shown in Figure 16. Compound tile indices are rendering directly into each quadrant of the texture memory using a *frame buffer object* (FBO).

preferred to embed the index data in the main texture object directly using mipmaps, when possible.

6. Rendering

In this section I discuss the details of the rendering logic used for this implementation. First, all required data structures required to store elevation data, compound tile data, and the vertex array used to define the terrain surface are allocated. Then, after binding the elevation data and index textures and enabling the custom vertex program, render the scene normally (using custom or fixed-function lighting) and then enable the custom fragment program which samples indices for images tiles and compound image tiles (and also exports latitude and longitude values per fragment). The output of the indexing render pass is directed to a FBO attached to a 32bpp texture of appropriate size, whose contents are then read to main memory and processed by the CPU for priority scoring. The pseudocode is given below.

```
Main()
        Allocate elevation data texture(s)
        Allocate $N$ compound image data textures
        Loop
              Bind elevation data textures
              Enable custom vertex program
              Bind $M$ compound image textures based
                    on priority scores
              If $shouldProcess()$
                    Enable indexing fragment program
                    Render scene
                    Fork:$processOutput()$
              Else
                    Enable lighting
                    Render scene
```

It should be noted that the fragment program for rasterizing tile indices to the framebuffer must sample several mipmap levels for each bound texture element and then combine the 8-bit values from each level to reconstruct the index (in this case, three 8-bit components are combined to produce a 24-bit index). The performance of this approach will vary on different graphics hardware implementations. The number of concurrent texture bindings reserved for compound tiles can be reduced to increase performance.

7. Processing Output

As shown in the pseudocode, a function is executed in order to process the index output. First, based on the current viewpoint location any unretrieved DTED tiles (the four tiles surrounding the current viewpoint) must be requested. I define a callback function, $loadElevation()$, that will be called as soon as the data is retrieved. Next, we iterate over the contents of the frame buffer and update four values: the minimum and maximum latitude and longitude. These values are used to define a bounding box which encloses the current view. This bounding box will then be used to query the OpenMap-based data service for the metadata for imagery tiles contained within the specified area. Then the priority scores for each compound tile index and each tile index referenced by the texel are updated based on the view priority function. The pixel score values for each index are combined to compute a total score per index. Pseudocode for this is process is given below.

```
ProcessOutput()
            Request needed elevation frames
                  Callback:$loadElevation()$
            Compute lat/lon bounds
            Compute index priority scores (image tiles and
                   compound image tiles)
            Execute coverage query (based on lat/lon bounds,
                  returns metadata)
            Define indices for new tile metadata
            Create new compound tiles metadata and define indices
            Render new compound tile indices into elevation index texture
            Update compound tile assignments based on priority
            Render new tile indices into assigned compound tiles
```

Request prioritized list of image tiles

Callback: $loadTile()$

After computing all index scores, I request the image tile metadata within the computed latitude and longitude bounds. Any new tiles are assigned a unique index. New compound tile metadata and indices are defined for any new tiles that are not contained by an existing compound tile.

8. Prioritization functions

The purpose of the prioritization function is to determine an indexed data element's relevance to the current view. Because the input to the priority function is the screen location of a reference in the framebuffer output, there are two primary attributes with which to work: depth and screen location. In order to account for screen coverage, I compute the sum of the depth and location terms for each pixel referencing a particular index. A generic form of the basic prioritization function is shown in 4.1.

$$P_d = \sum_{i}^{n} (2 - Depth_{d_i}) \times Location(d_i) \tag{4.1}$$

Since depth values are on the range [0,1], I subtract the depth from the scalar value 2. This ensures that all pixel contributions are counted, but data element references closest to the near clipping plane are weighted twice as much as those furthest away. Finally, I define an additional scaling function ($Location(di)$) that can, for example, increase prioritization values near the center of the view. I have used a simple location prioritization function defining an elliptical region (since terrain is often viewed along a horizon) with little observed difference. However this function could also be used to define clipping areas if data near the screen boundaries is not desired, or to interactively define regions of interest using the mouse pointer.

a. Memory Management

A simple memory strategy is used to manage video memory. First, compound texture assignments are prioritized to ensure video memory is allocated to the most important areas of the current view. This assignment simply links a pre-allocated block of video memory with specific compound tile metadata that defines the latitude and longitude coverage bounds of the memory assignment. Once a compound tile is assigned allocated memory, tile indices can be loaded into mipmap memory.

Next, the binding of the allocated compound textures are prioritized since there is a finite amount of addressable (bindable) video memory (both in terms of bytes and number of texture units). In my experience with large textures, the bindable byte limit is usually hit before the number of textures (usually 32). Note that only bound compound textures can contribute their tile indices to the scene, therefore this set defines the potential data elements that can be identified in the resulting priority list without modifying texture binding assignments. This implementation avoids unnecessary changes in these assignments since the entire terrain geometry is defined by a single generic vertex array that can be rendered quickly (changes in texture binding are not possible during the rendering of a vertex array). An example of the visualization application that highlights this texture assignment behavior is shown in Figure 20.

I maintain a table mapping compound tile metadata to texture allocation IDs. The top A compound tiles, ranked by priority, will have table mappings to allocated video memory. Whenever a compound tile allocation is updated, tile indices must be repopulated into the mipmap memory. I maintain another table that maps compound tiles to bindable texture locations. Priority scoring ensures that the top B compound tiles are bound (i.e., addressable), where B is defined by subtracting the number of texture elements used for

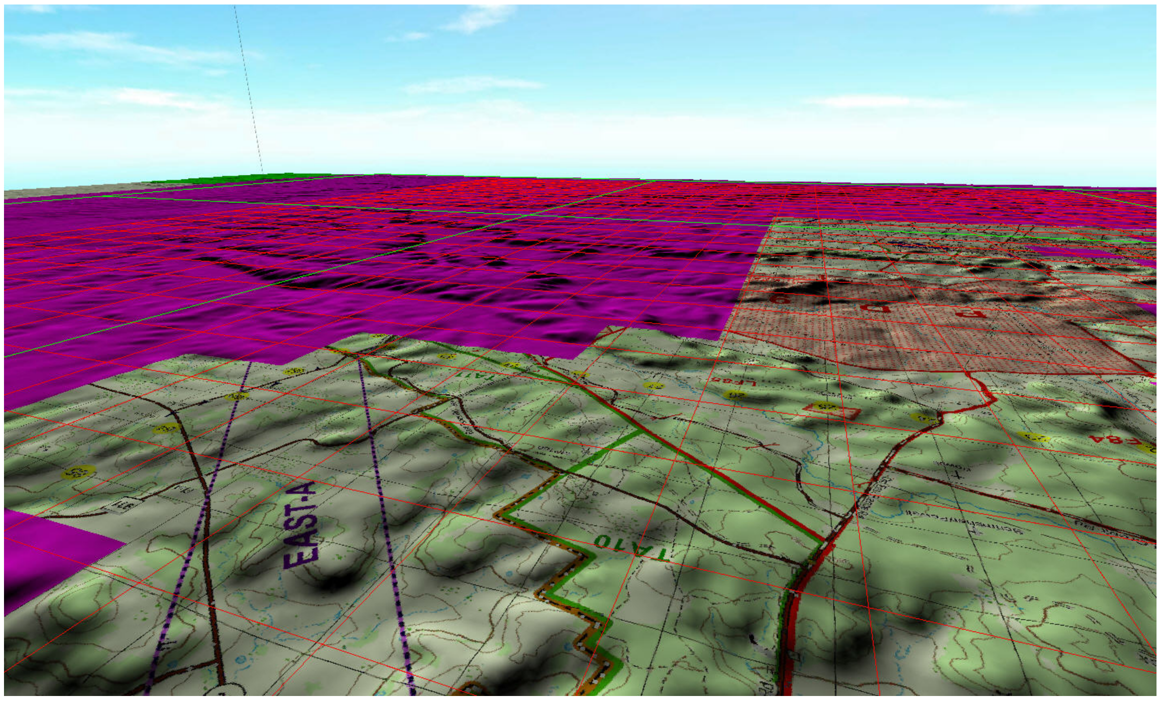


Fig. 20. In this example the application is currently retrieving the set of viewable imagery tiles for a cartographic map over the viewable area. The bounds for current compound tiles are drawn in red and the bounds for individual tiles are drawn in green.

elevation data and indices (three in this implementation; one for the elevation data, one to store compound tile indices aligned with this elevation data, and one for the skybox texture) from the maximum number of texture binding locations. In this case, $B = 32 - 3 = 29$. The implementation of these table mappings ensures that changes in texture allocations and bindings are minimized.

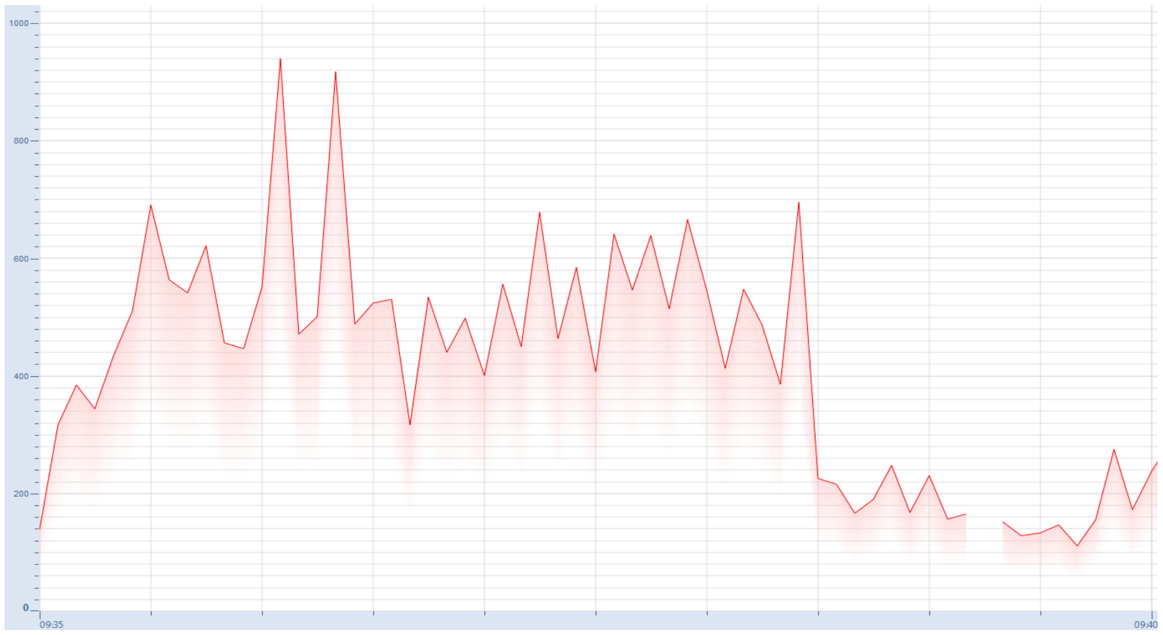
D. Results

I have tested this implementation using a variety of combinations of elevation and imagery resolutions. I have used both DTED1 and DTED2 elevation data, and RPF map sets ranging from 1:25K scale cartographic maps to 1:1m satellite imagery. Also, I have tested the performance of the terrain rendering application using both local and remote data services, where remote data retrieval was conducted over both wired and wireless network connections. The test system was a Windows 7 laptop with a 3 GHz Intel Core2 Extreme CPU with 8 GB RAM and a nVidia QuadroFX 3700M graphics card with 1GB dedicated video memory (addressable) and approximately 4GB shared video memory (virtual). We observed stable rates of around 25 fps when using DTED1 terrain resolution (approximately 1.4 million vertices) and around 13 fps when using DTED2 (approximately 13 million vertices). I will discuss specific performance results in more detail.

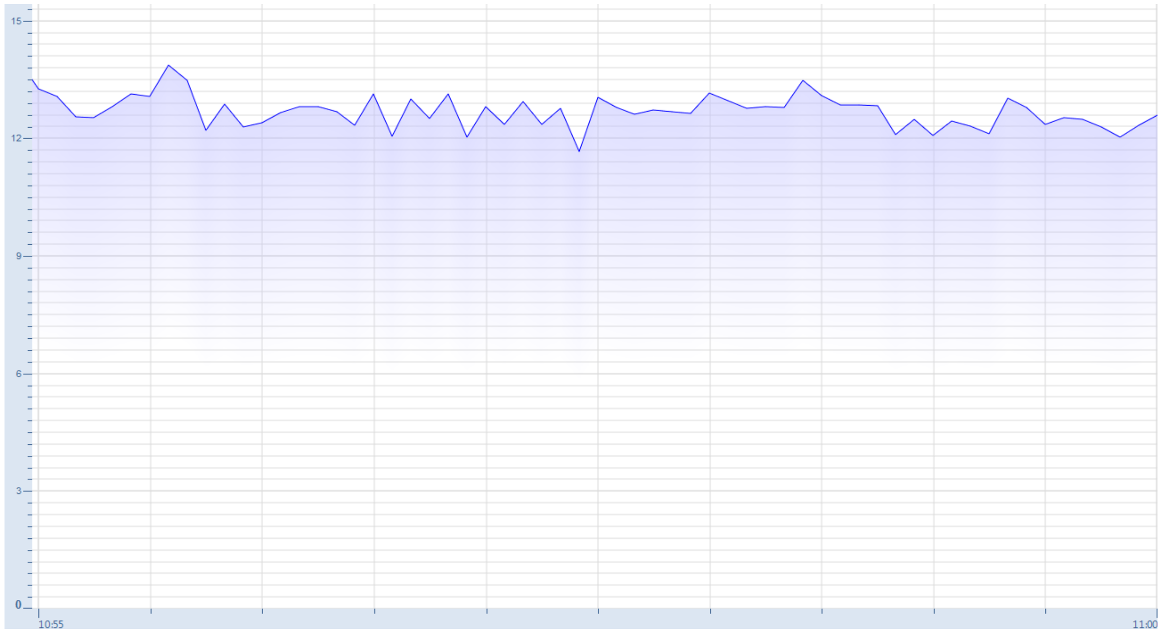
1. Performance

In order to evaluate the effectiveness of this approach, I have collected performance metrics during several controlled executions of the terrain visualization implementation. In each case I have varied the latency of data retrieval, the amount of geometry in the scene, or the resolution of the terrain imagery and I have captured the framerate and texture loadrate of the application. The evaluation of these results supports my initial hypothesis that use of asynchronous view-prioritized data retrieval can enable stable rendering performance regardless of the size or latency of retrieval of a remote dataset.

The metrics recorded during the first experimental evaluation are shown in Figures 21–23 and include the retrieval latency and framerate of the terrain visualization application.The retrieval latency is defined as the time in milliseconds for each individual imagery tile request to be completed. Both values are shown as averages over five-second periods.

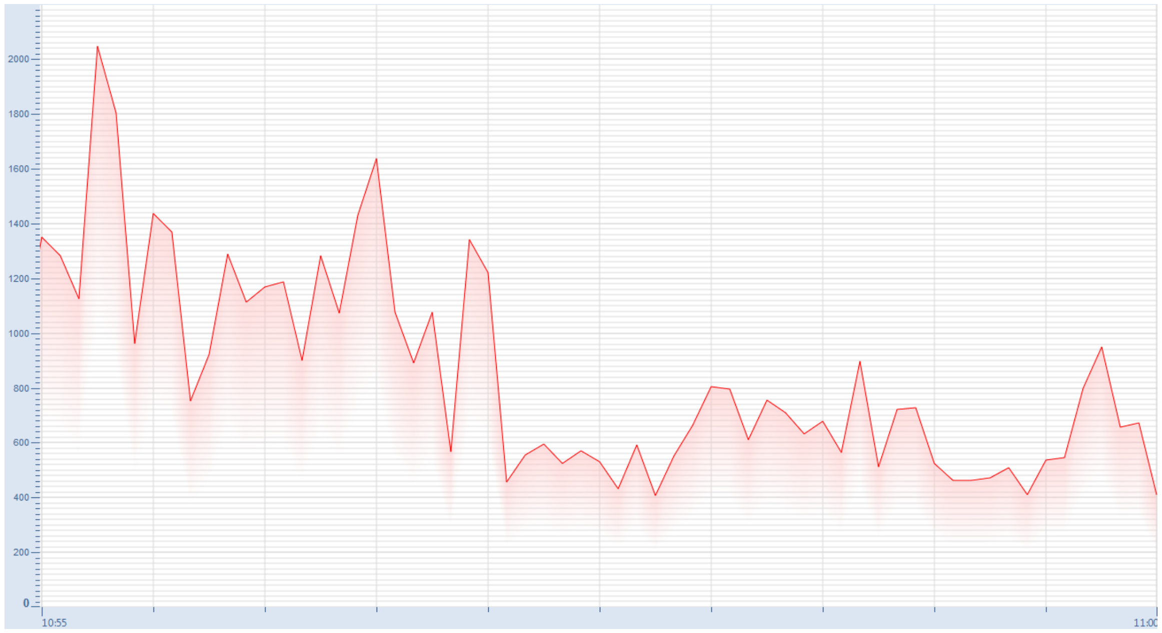


(a)

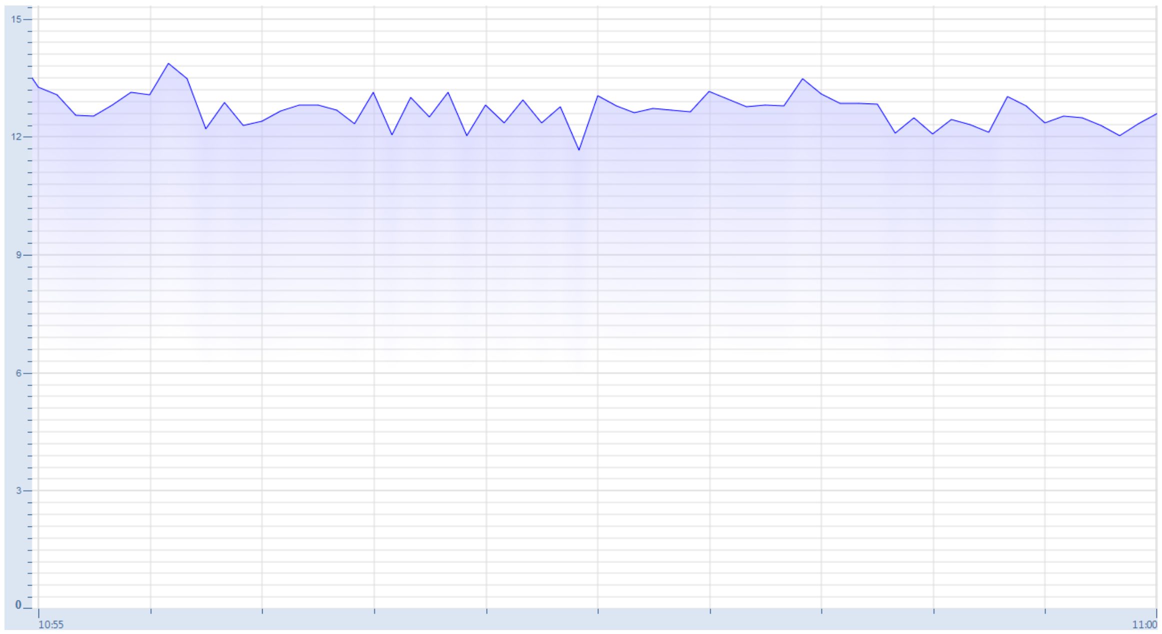


(b)

Fig. 21. In this experiment I vary the latency of tile retrieval by querying a local database. Each graph shows five minutes of performance metrics collected from the terrain visualization application. The tile retrieval latency is shown in red (a) and the framerate is shown in blue (b). The *x*-axis of the graph depicts time and the *y*-axis depicts the scalar values of the framerate (1/ms) and data retrieval latency (ms). The break in the top graph indicates a period where no data was retrieved.

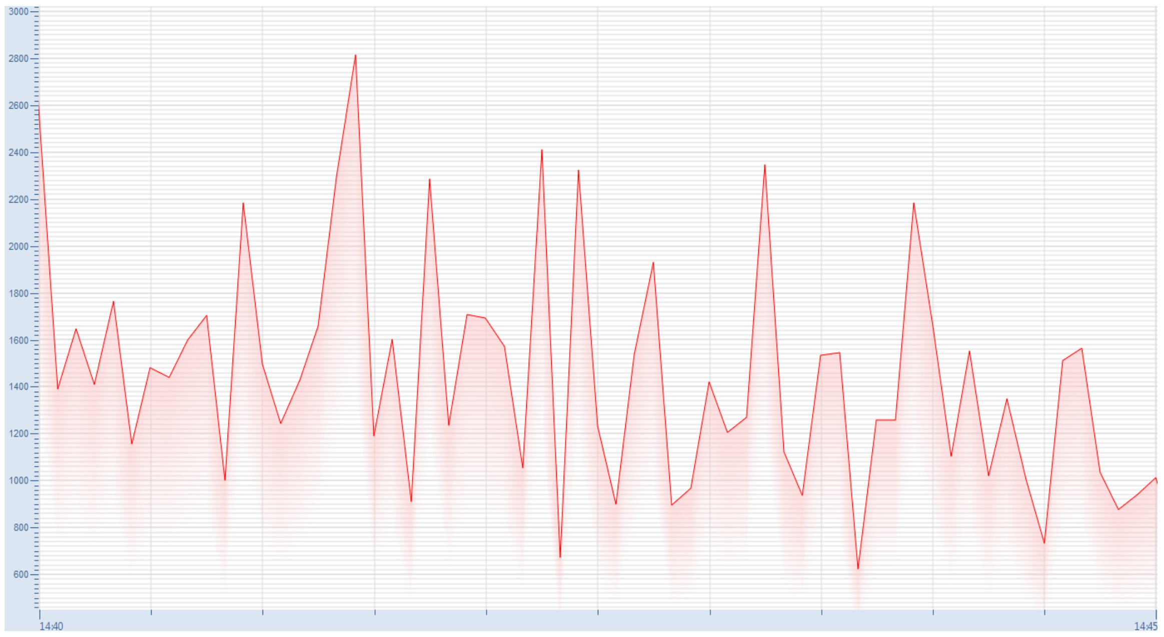


(a)

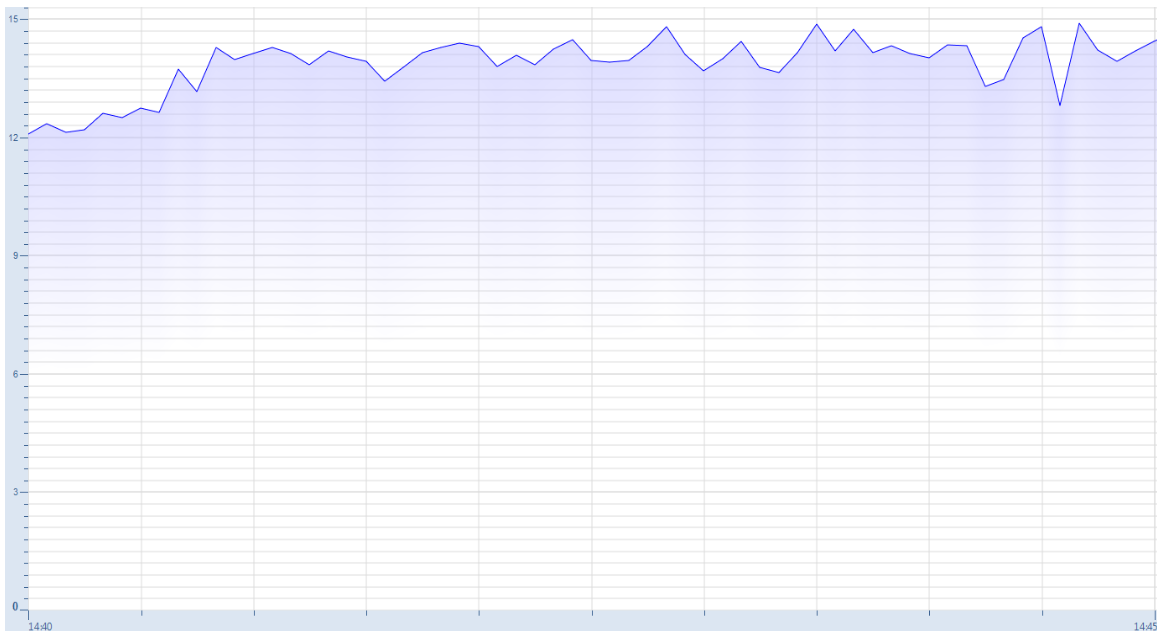


(b)

Fig. 22. In this experiment I vary the latency of tile retrieval by querying a remote database accessed via an Ethernet LAN. The tile retrieval latency is shown in red (a) and the framerate is shown in blue (b).



(a)



(b)

Fig. 23. In this experiment I vary the latency of tile retrieval by querying a remote databases accessed via a network with artificial latency. The tile retrieval latency is shown in red (a) and the framerate is shown in blue (b).

During this experiment the location, map set, and terrain surface resolution remained constant. The application was configured to render DTED2 resolution elevation data (approximately 30 meter elevation postings, 3600 postings per degree) from an area near Kandahar, Afghanistan using a 1:25K scale topographic line map. In this experiment, the local dataset consisted of approximately 2 GB of imagery data on a local hard drive, while the networked datasets were queried via a custom socket application running on a remote system with over 200 GB of total imagery data.

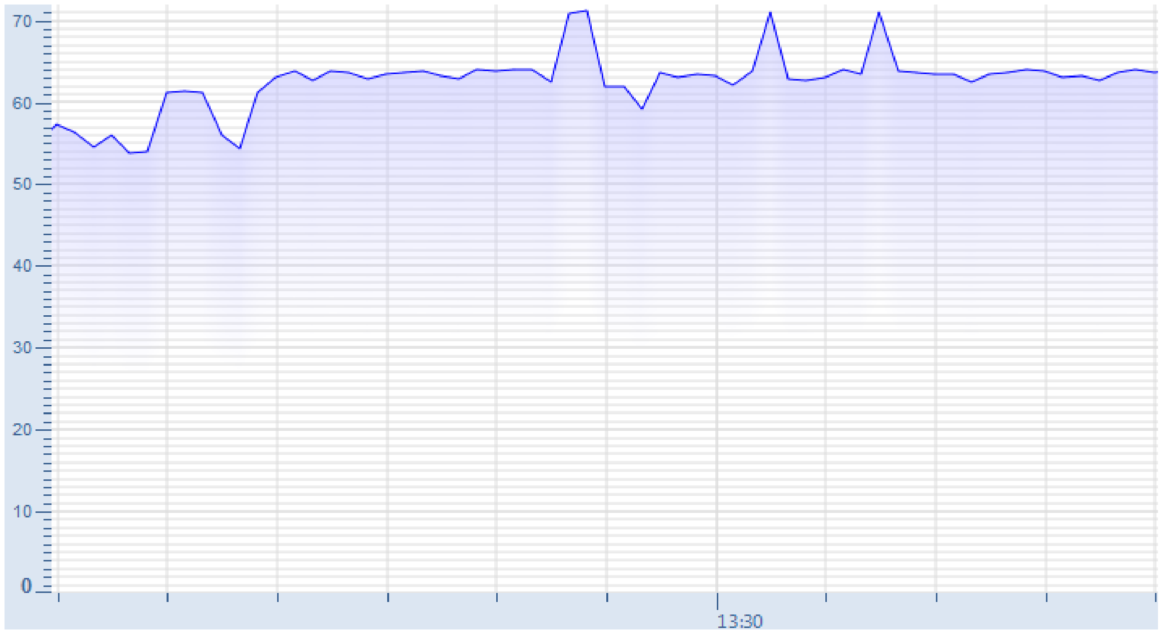
While we might expect the overall retrieval latency to increase when querying a larger remote database for imagery tiles, in this experiment I observed that the latency of the local database was comparable to the latency of the remote database. This is most likely because the local database must compete with the visualization application itself for local computational resources while the remote queries are executed by a dedicated database system. An additional experiment introduced artificial delay with moderate variance (the artificial delay system was configured to introduce a one-second delay to each transmission with a 100ms variance). The experimental results are shown in Table II.

Table II. Latency Experiment Results

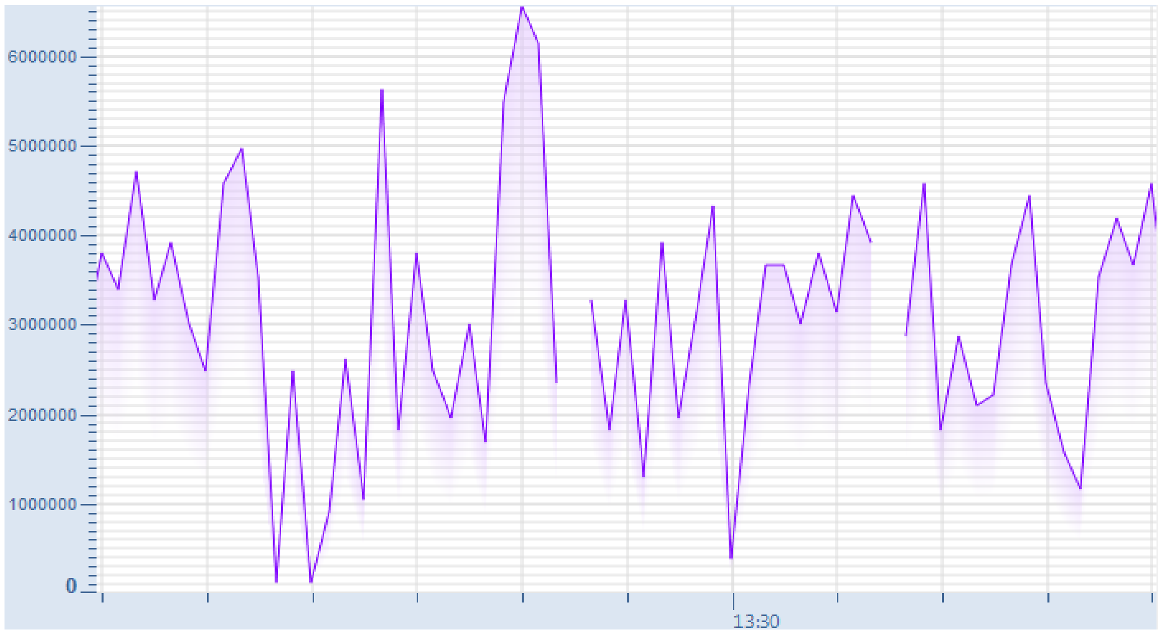
	Latency (ms)	Framerate (1/ms)	Pearson (%)
Local Disk	418	12.49	-0.158
LAN	854	12.69	0.154
Artificial	1452	13.88	-0.207
Total	1159	13.47	0.343

The table also includes the Pearson product-moment correlation coefficient, *r*, computed from the experimental data. The Pearson coefficient is defined on a range $[-1, 1]$ and measures the degree of linear correlation between two observed variables, where a measure of zero indicates no correlation. For the purposes of this work, I expect to observe little or no correlation between the observed variables, and therefore expect to observe r values whose absolute value is less than 0.3. I computed averages and r values within each experiment as well as between all experiments (designated by the last row in the table). The results of this experiment varying latency showed a higher than expected positive correlation, prompting further experimentation. To this end I conducted two additional experiments aimed at exploring the nature of this observed correlation. These experiments were designed to test correlation between the framerate and loadrate (rate at which texture data is loaded on the GPU), since such a correlation could be responsible for the high r value observed in the first experiment.

In this next experiment I varied only the amount of geometry used to render the terrain surface. In this case the application was configured to visualize 1-meter satellite imagery from Aberdeen Proving Grounds (located near Aberdeen, MD) using both DTED1 and DTED2 grid resolutions. During the experiment I collected the framerate of the graphics hardware as well as the *loadrate*, as shown in Figures 24–25.

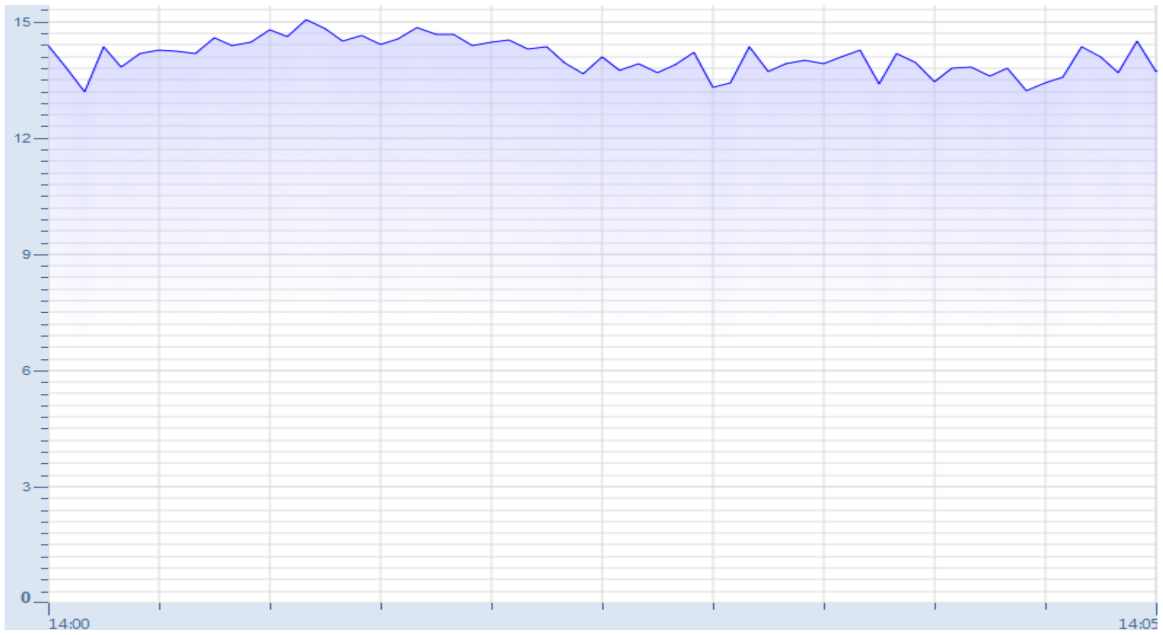


(a)

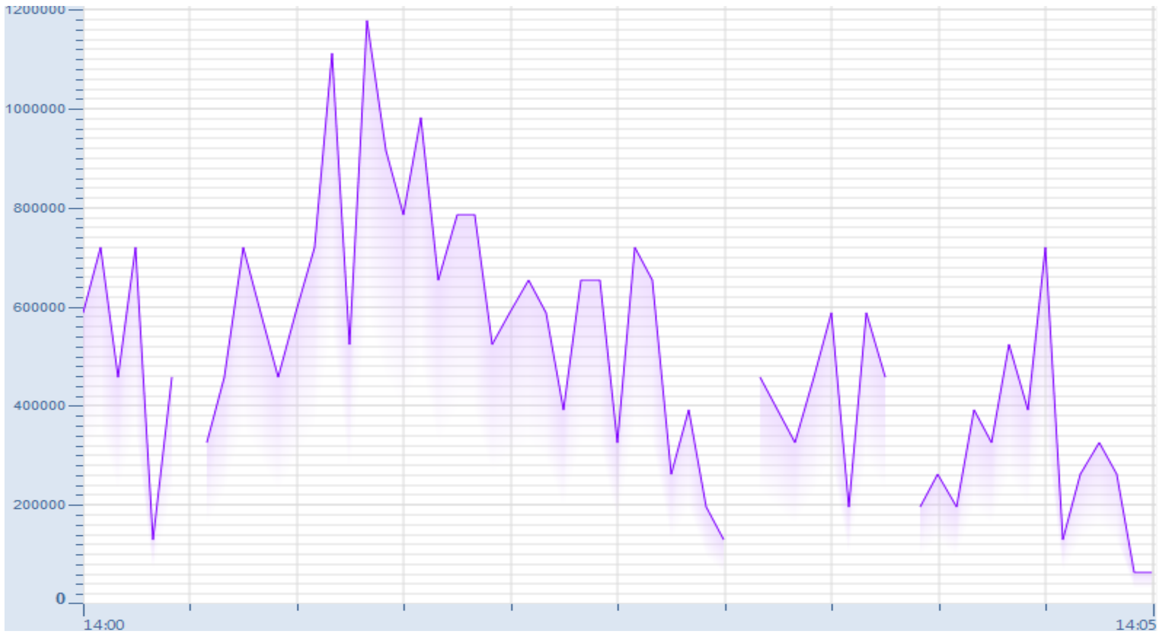


(b)

Fig. 24. In this experiment I rendered a DTED1 resolution terrain grid (approximately 1.4 million vertices). The framerate is shown in blue (a), and the loadrate is shown in purple (b).



(a)



(b)

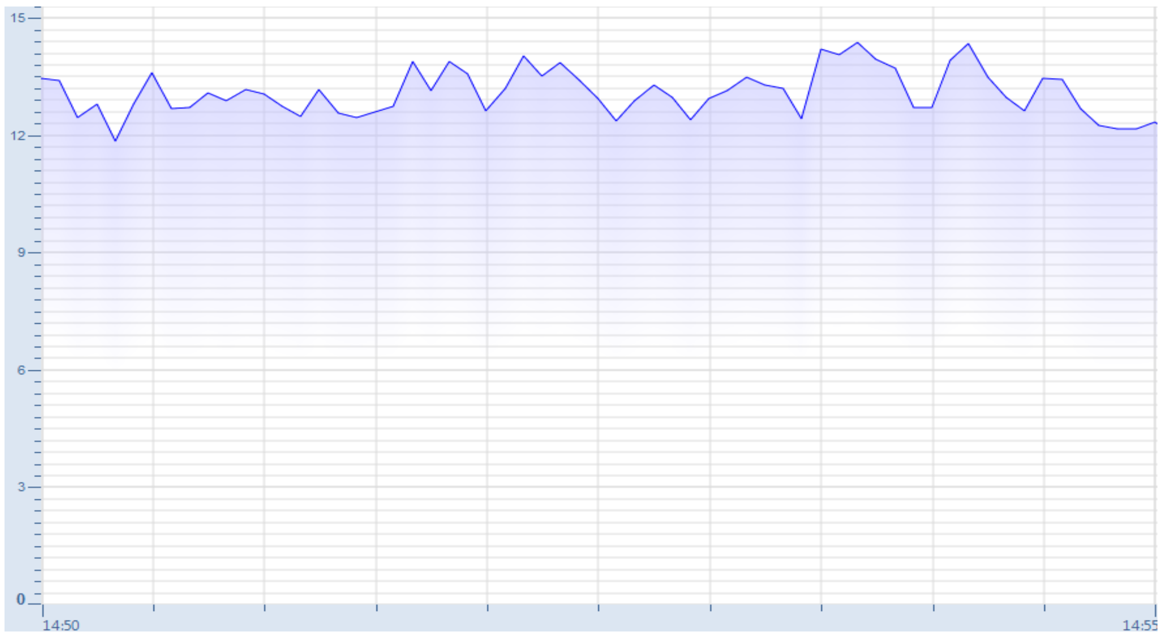
Fig. 25. In this experiment I rendered a DTED2 resolution terrain grid (approximately 13 million vertices). The framerate is shown in blue (a), and the loadrate is shown in purple (b). The breaks in the bottom graph indicate periods in which all the data for the current scene had been retrieved, and therefore no data was loaded.

Loadrate is defined as the number of bytes of texture data loaded into graphics hardware memory per time unit. I observed that while the framerate was significantly lower when rendering DTED2 resolution (approximately 13 million vertices as opposed to 1.4 million for DTED1), there was no significant effect on the loadrate within experiments. However the results, shown in Table III, demonstrate that between experiments there is a noticable positive correlation between the framerate and the loadrate. This can be explained by understanding the operation of the graphics hardware. Since current hardware implementations do not allow rendering and texture loading operations to be executed in parallel, an increase in the time spent rendering (i.e., lower framerate) leaves less time available for texture loading. In this case, the results are acceptable since the outcome favors rendering performance, which is our goal in this interactive visualization application. However, when large amounts of texture data saturate the loading routine, the result can adversely effect the rendering performance. To this end, I have introduced the use of *load throttling* to balance the time spent loading texture data with interactive framerates and tested this approach in a separate experiment.

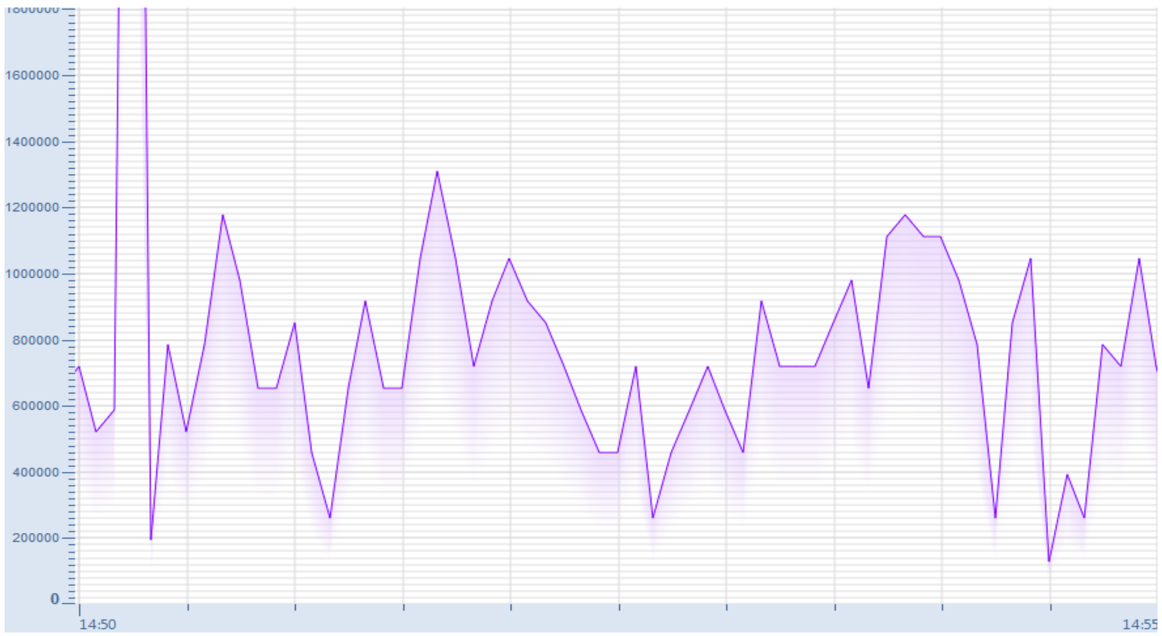
Table III. Geometry Experiment Results

	Framerate (1/ms)	Loadrate (kB)	Pearson (%)
DTED1	62.67	3107961	-0.018
DTED2	12.40	557056	0.194
Total	38.39	1815403	0.786

In the final experiment I varied the resolution of the terrain imagery data while again rendering a DTED2 resolution elevation grid located near Kandahar, Afghanistan. The results of this evaluation are shown in Figures 26–27 and include the framerate and loadrate when using a 1:25K TLM and 1-meter satellite imagery.

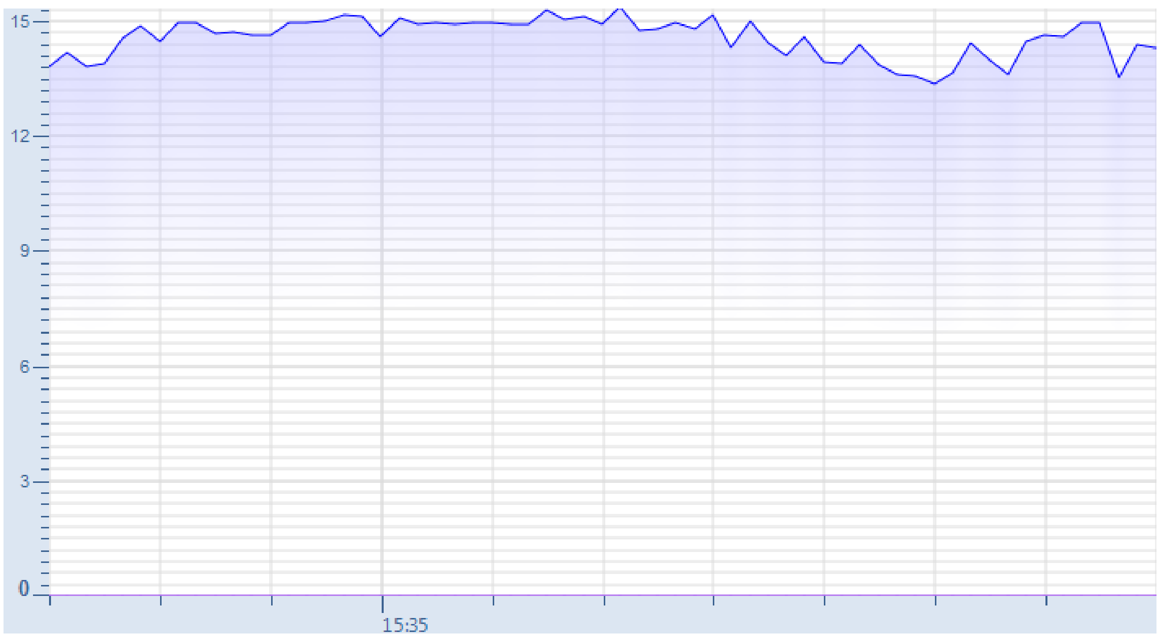


(a)

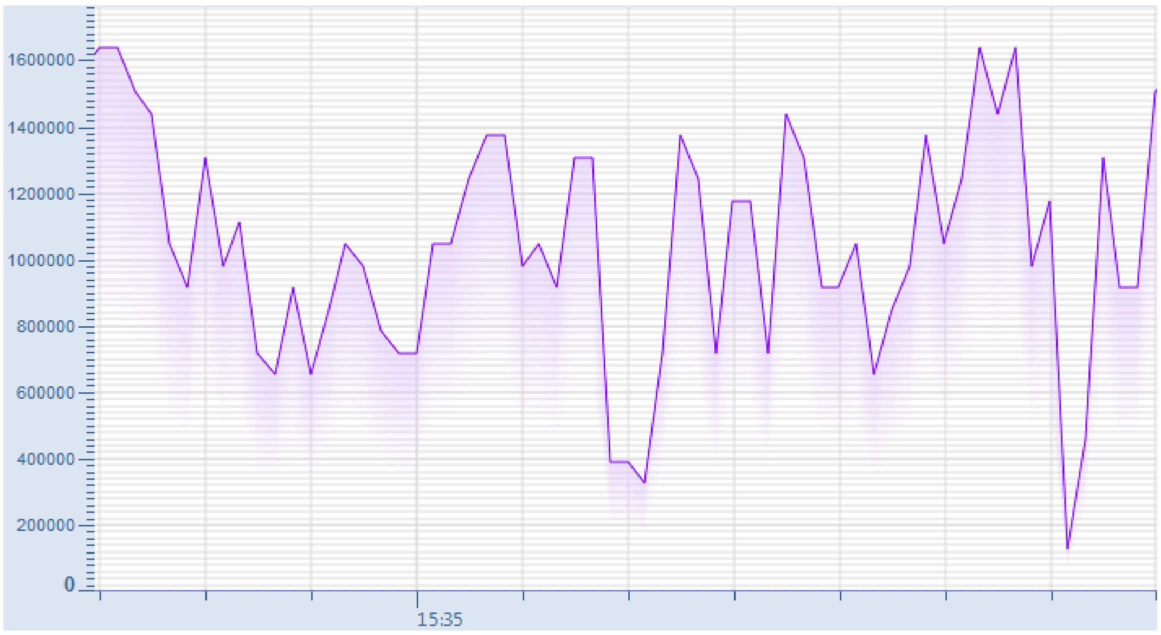


(b)

Fig. 26. In this experiment I rendered a 1:25K resolution topographic line map (TLM). The framerate is shown in blue (a), and the loadrate is shown in purple (b).



(a)



(b)

Fig. 27. In this experiment I rendered 1:1m resolution satellite imagery. The framerate is shown in blue (a), and the loadrate is shown in purple (b).

For comparison, the TLM map set contains several hundred image tiles, while the 1-meter imagery map set contains several thousand tiles. These results show that the average framerate and loadrate of the system are not significantly affected by the resolution or tile count of the remote dataset when using low-resolution data. Initially I did observe a potential correlation between framerate and loadrate when using high-resolution data. As discussed, this is because rendering and loading operations are mutually exclusive and therefore must be executed serially by the graphics hardware. The proposed *load throttling* implementation amortizes the cost of loading texture data over a longer period of time, similar to the "simulation amortization" method introduced by Harris et al. [27] to spread the computational cost of a fluid simulation used to render interactive cloud formation on the GPU. The method I have implemented will allow only a specified amount of time (300ms in my implementation) to be consumed by loading operations every second. As we can see from the results shown in Table IV, throttling the texture loading for high-resolution imagery produces the expected low Pearson value by removing the dependency between framerate and loadrate.

Table IV. Resolution Experiment Results

	Loadrate (kB)	Framerate (1/ms)	Pearson (%)
1:25K TLM	816514	13.11	-0.006
1m Imagery	2127234	14.01	0.637
1m w/ Throttling	1041055	14.58	-0.3678
Total	1471874	13.56	0.597249
Total w/ Throttling	928784	13.84	0.09615

These results indicate that the basic principle of the proposed method that asynchronous data retrieval enabled by indexing pre-allocated memory coupled with efficient view-prioritization improves interactive performance when working with large remote datasets is sound. However, in order to effectively implement this method on current graphics hardware certain issues must be addressed. Specifically, since current graphics hardware implementations do not allow rendering operations to be performed in parallel with texture loading operations, *load throttling* must be enforced to ensure that data loading operations do not saturate the graphics pipeline, preventing interactive rendering performance. This is a constraint introduced not by the proposed method, but by limitations of current graphics hardware.

E. Ongoing Work

I am currently testing this application in several data analysis scenarios using domain experts, including spatial analysis of high-resolution tracking systems involving large numbers (several hundred) of individual sensors attached to players operating in close proximity. An example of this application is shown in Figure 28. The goal of this application is to provide detailed geospatial context to aid in the analysis of wireless network performance over varying terrain conditions, and therefore small-scale terrain features are important within the interactive visual analysis environment. I would also like to explore the impact of large imagery datasets that are updated on-the-fly. This will require the data service to inform the visualization application when certain tile indices have updated data so that the texture memory locations can be marked to be updated.

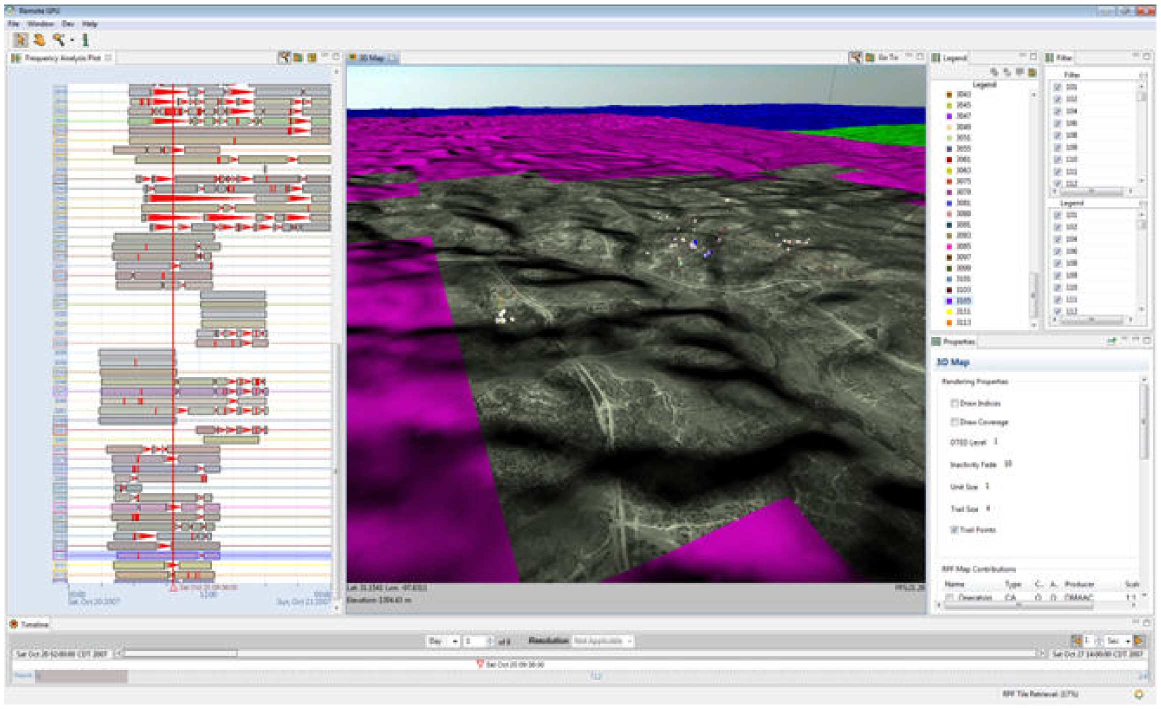


Fig. 28. Example of our terrain rendering application embedding within a coordinated multiple view visualization environment. Individual sensors attached to players operating in close proximity are rendered as colored spheres on top of the terrain, which is textured with 1m satellite imagery. The density of position updates per sensor are shown on the left. A legend and timeline are included on the right and bottom, respectively. The magenta area of the terrain shows where compound tile memory is currently allocated and bound. The green and blue areas are used to distinguish different frames of elevation data (stored in separate quadrants of the allocated texture memory.

CHAPTER V

APPLICATIONS

In this chapter I will discuss several novel applications developed and delivered to users using the proposed framework. First, I will discuss a custom glyph-based visualization developed for a specific group of users responsible for validating large collections of sensor data. I will discuss the details of this technique, and results of several user studies and expert opinion. Next I present a custom spatial visualization implemented to enable analysts to measure the accuracy of situational awareness (SA) data in a wireless network, as well as expert opinion supporting the advantages of this technique.

A. Glyph Visualization for Temporal Analysis of Spatial Sensor Activity

As distributed sensor networks become more common, methods that enable quick recognition and proficient analysis of pattern irregularities from these sensors will become more important. This work aims to improve visual recognition of temporal patterns within a network of spatial sensors. I will present a new visualization technique that we have verified in a real-world analysis application designed to enable analysts to quickly inspect large collections of data for critical failures.

This work can be applied to other applications, but is inspired by interaction with colleagues who collect and analyze large datasets produced by networks of spatial sensors. The goal for these users was to ensure accurate performance of each spatial sensor as well as complete collection of the data produced by the sensors. The sensors are attached to players or vehicles participating in a simulation, and are designed to report their location at a fixed rate when at rest, or more often when the sensor is moved. The challenge for these users was to provide a quick and thorough overview of daily recorded sensor activity with emphasis on highlighting anomalous behavior. Anomalies defined by the domain experts were primarily temporal (e.g., a sensor failing to report within the expected time), however our technique enabled the domain experts to detect both spatial and temporal anomalies. An example of the visualization application is shown in Figure 29.

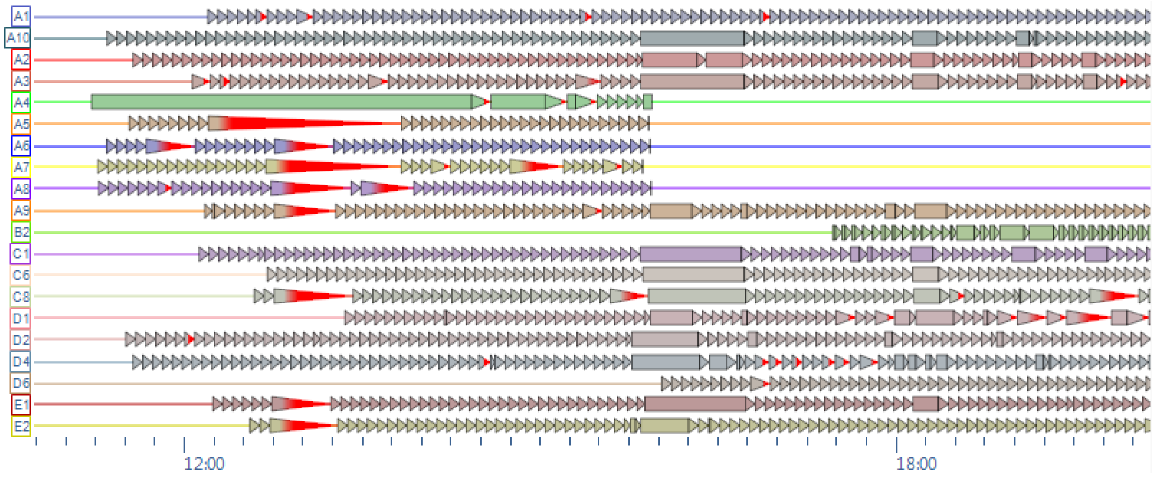


Fig. 29. Temporal activity patterns exhibited by a network of spatial sensors. Each row depicts the sensor reporting activity for a unique sensor, and each triangle represents a period of time between reports. Five minute threshold violations are highlighted in red, and triangles less than 3 pixels in width are merged to form quadrilaterals. A coordinated violation is shown by seven of the sensors in the beginning of the time period shown. The sensors used in this work increase activity when in motion, therefore in this example the quad areas represent spatial movement. A coordinated movement involving at least twelve sensors is shown in the middle of the period depicted, at the same time that five sensors stop reporting unexpectedly.

During interviews with domain experts, we were able to identify several specific analysis tasks which we sought specifically to optimize. Foremost, users should be able to quickly and accurately identify failure patterns within the sensor network (i.e., when expected sensor activity is not present), and investigate the temporal and spatial context of these failures. Second, users should be able to easily recognize expected patterns such as coordinated movements and start/end times of reporting periods to validate the accuracy and completeness of the sensor data. Finally, users should be able to identify and investigate the context of spatial reporting anomalies to support analysis of sensor problems. We designed a visualization method with these user-defined analysis goals in mind, and likewise the tasks in the user studies are designed to evaluate the effectiveness of using this information visualization technique when executing these types of analysis tasks.

In addition to traditional bar charts and line graphs, analysts had been using a row-based state plot in which time periods (at a preset five-minute resolution) were colored green or red based on whether or not sensor activity occurred. An example of this technique implemented within the ProDV software environment is shown in Figure 30.

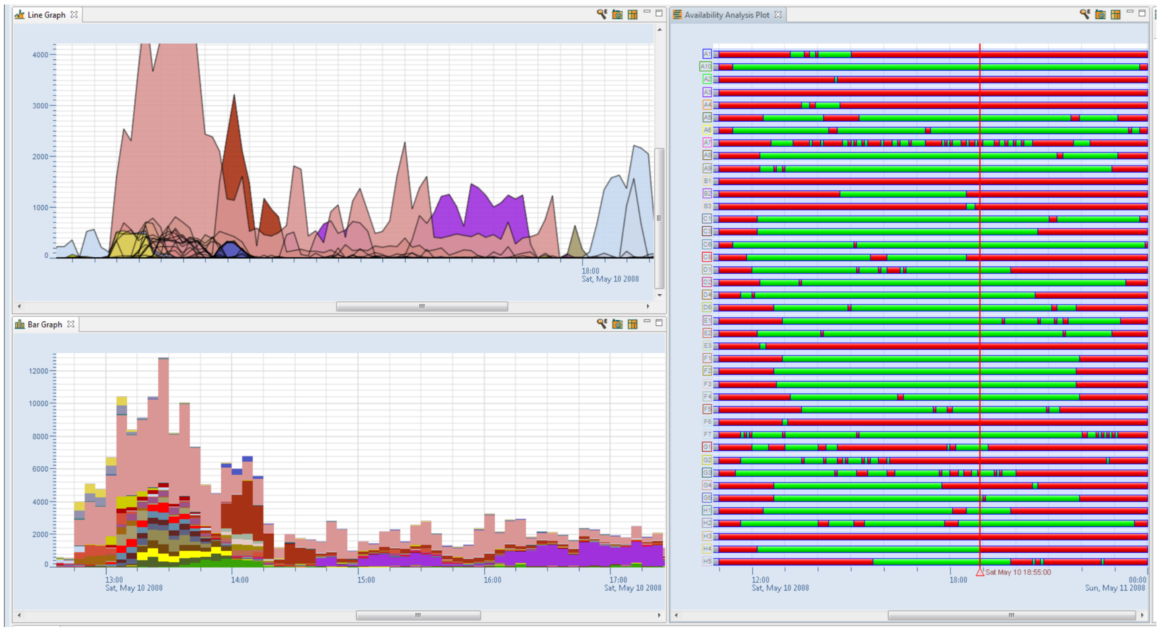


Fig. 30. Example of existing methods similar in capability to our domain experts' previous analysis technique. Line and bar graphs on the right show the total number of position updates sent by each sensor for each five-minute interval. The chart on the left shows a row for each sensor, and is colored green for each five-minute period in which a position update was recorded, and red otherwise.

While these techniques were sufficient to identify a temporal sensor failure, the process to produce these static visualizations was prone to error and the resulting static visualization became cluttered when displaying multiple sensors. Also, temporal pattern recognition was not easily accomplished using such methods. Our approach was to design and implement a custom interactive visualization and provide essential visual analysis features such as rescaling, filtering, and querying for details [64]. I will describe the glyph-based visualization technique we developed to facilitate analysis of these sensor activity patterns. I will also discuss evaluation of this technique using user studies and expert opinion.

1. Proposed Method

While the techniques shown in Figure 30 can display the primary attributes of the data - the number of reports per time unit and the sensor ID - they do not allow the analyst to effectively discern changes in temporal sensor activity patterns due to screen clutter and data aggregation. These methods also do not effectively highlight attributes derived from the data: the individual temporal anomalies caused by sensor failures and the exact time they occurred. Sensor failures are defined as a failure to generate a position update within the expected time. Five minutes was usually the expected value for our users, although in practice the value is user-specified (depending on the sensor type used).

In the following sections I will discuss the general format of our input data, the detailed rendering techniques used to create the visualization, and the user interactions provided by the application to facilitate visual analysis.

a. Input Data

The input data is comprised of an ordered sequence of location reports for a collection of sensors, and each sensor is assigned a unique identifier. Each sensor report includes its ID, a timestamp, and a geospatial coordinate. Records with duplicate IDs and timestamps are ignored. For our particular data set, times were given in seconds, which was sufficient for our analysis. The datasets our domain experts were interested in included anywhere from 30 to 100 sensors, although more recent events have included several hundred individual sensors.

b. Design Rationale

In the design of the visualization method, several considerations led to the final design discussed here. We chose to use a triangle to represent periods of time between sensor reports because:

- It allows us to clearly distinguish when the event occurred (with the vertical base of a triangle).
- In contrast to a single mark, it has volume proportional to the length of time between reports, and is thus easier to comprehend visually.
- In contrast to a rectangle (which also provides that visual area), the triangle shape allows one to clearly see the beginning point when juxtaposed with the tip of an adjacent triangle.
- The shape is very simple, making it much easier to comprehend over a larger region then more complex glyph shapes that could potentially convey more information individually.

We also decided to visually aggregate small triangles into rectangles because:

- The limitations in pixel-based display make it difficult to even recognize a triangle with a width of two pixels, and impossible with one.

- The rectangle clearly indicates rapid updates in a period of time (“rapid” being relative to the scale being displayed, of course).
- We can still employ red highlighting to indicate critical features (e.g., failures) within the rectangle.

c. Rendering Details

The visualization is rendered in rows of equal height, one row for each unique sensor ID. A horizontally-oriented isosceles triangle is drawn in between each sensor report, where the horizontal axis references time. A vertical line marks the position of the last sensor report in each row. If the horizontal length of any triangle, which corresponds to the length of time between sensor reports, is less than some minimum width e_{screen} at the current zoom level, then a quadrilateral is formed so that neighboring small triangles can be merged to form a solid rectangle. The reasoning for this is that a triangle or group of triangles that are less than e_{screen} will be difficult to discern and will visually approximate a solid rectangle anyway. We use five pixels as a default for e_{screen}, however since there is variation in pixel size among displays this value is adjustable. An example of these glyphs is shown in detail in Figure 31.

The algorithm for drawing each row is as follows:

```
For each sensor

    For each report, compute $x_i$ (the exact horizontal
    position based on view scale).

    If there is no next report, just draw a vertical
    line. Otherwise, compute $x_{i+1}$.
```

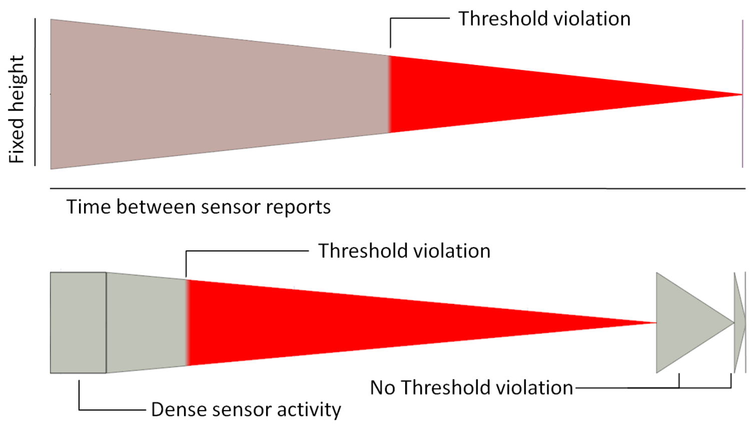


Fig. 31. Details of our row-based glyph rendering method. The colors of each row are user-specified.

If $x_{i+1} - x_i < e_{screen}$, begin a new quad or continue the current quad. Otherwise, draw the triangle.

Draw threshold violation, if appropriate.

Threshold violations occur whenever a period between sensor reports is longer than the user-defined threshold (default is five minutes, based on majority of equipment used by our domain experts). These threshold violations are drawn using at least e_{screen} columns of red pixels at the appropriate point on the triangle or rectangle. By drawing at least this many columns of red pixels, we ensure that at any scale threshold violations will be distinguishable. This is illustrated in Figure 32. If the user does zoom out until such a triangle is less than e_{screen} pixels wide in the current screen projection, then we assume the user is not interested in this level of detail and allow the triangle to merge with any neighbors (which may or may not be in violation of the threshold). Regardless of scale, however, any pixel column which contains a threshold violation will always be highlighted to ensure that violations are always visible.

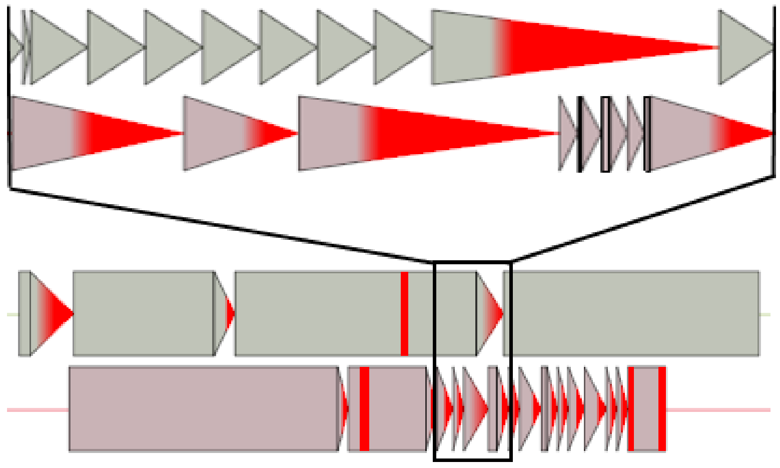


Fig. 32. Example of the proposed method at different scales. A one hour view of the two sensors' activity is shown at the top, a twelve hour view is shown at the bottom. The one hour view depicts the time period outlined in the twelve hour view.

We render each row of the data using the given algorithm at the current zoom level and for the currently specified time window, which is by default one day in length. The row height can also be adjusted by the user to ensure that the graph is readable on the current display.

In regard to color, we allow the visualization environment to assign what were essentially unique but random colors chosen from a finite color scheme to the set of sensor IDs. However, within this visualization we desaturate the assigned sensor color when drawing the rows to ensure a high contrast with the red threshold violation indicator. The true assigned color is used to draw the line for each row (visible before the first and after the last sensor report), and for the sensor ID text drawn in the margin. This allows the visualization to properly highlight the temporal threshold violations while still preserving the aesthetic attributes of each sensor ID, which in our visualization environment are shared between multiple visualizations. In practice users are encouraged to manually assign color groups because of the potentially large number of sensors. We did investigate use of some existing color schemes, such as those proposed by Brewer [28]. However these schemes were not designed to accommodate large numbers of distinct entities (the largest ColorBrewer scheme has only 12 distinct colors designated). For evaluation I ensured that all users were presented the same essentially random color scheme, since I am not attempting to evaluate the effect of color in this work.

d. Interaction

The interactive environment provides select, pan, zoom, and filter operations as well as detailed information on demand. Since this is a row-based visualization, we provide rescaling along the x axis (time). Additionally, the user can set the view to one of several preset scales: hour, day, or week.

The glyph-based visualization can be (and is, in this implementation) linked to a map displaying spatial information associated with the sensors. Specifically, for a particular time, the position of the sensor at its most recent report (or interpolated between reports) can be shown. The map can also be configured to display a user-specified number of past positions to define a trail that depicts a sensor's spatial path.

Row selection is indicated by highlighting and is linked to a common legend. Users can select multiple rows and selections can be grouped to cause them to be collocated in the visualization. The interactive map will recenter to focus on the location of a selected sensor.

The details provided by hovering and clicking the mouse on any part of the visualization window include the exact times of the neighboring sensor reports, the exact length of time between sensor reports, and the sensor ID.

Since the visualization environment provides temporal animation features, we also included the ability to automatically pan the view during animation. This allows the user to observe in detail the reporting pattern of a sensor as its position is animated on the map. Since domain experts had an informed expectation of sensor activity during various spatial movements, this feature allowed them to quickly identify unexpected behaviors.

2. Implementation

The proposed method was implemented using ProDV, the previously discussed visual analytics framework for interactive data visualization applications [55]. Visualizations are implemented using OpenGL, and common controls such as a timeline, legend, and filter are provided. As discussed, this framework facilitates rapid development of coordinated multiple view (CMV) type visualizations, which I have taken advantage of in order to compare the use of combinations of visualization techniques to perform specific analysis tasks.

a. Input Data Format

Data was provided in text files which were exported after aggregating processed log file data from each sensor. Each text file contained the data for all sensors for a complete day. Depending on the amount of activity in a day, each file could contain millions of sensor reports and be hundreds of megabytes in size. While the input data files did contain more information than just the time and location of the sensor (e.g., speed or orientation), the proposed glyph used only the primary temporal and spatial components of each sensor report.

3. Evaluation

In addition to iterative and informal evaluations with expert users, which unfortunately did not afford the opportunity for formal evaluation efforts, I conducted two small user studies to evaluate the effectiveness of our approach. No incentive was offered for participation in the 90-minute study, and thus I recruited only a small number of volunteers in the approved time frame. All participants recruited were Computer Science graduate students specializing in graphics, all with moderate data analysis experience. First, I recruited a small group of graduate students to participate in a task-based evaluation of temporal pattern recognition using this method. This first study focused on use of this technique strictly for temporal analysis with no reference to spatial components of the data. Critical analysis tasks were identified by our domain experts and focused on identifying coordinated sensor failures, coordinated spatial activities, and other feature identification tasks (e.g., identify the sensor with the longest period of sustained activity). Next, I asked a group of volunteers with a broader range of analysis skill to participate in a similar task-based evaluation focused both on temporal and spatial analysis tasks. Spatial tasks focused on identifying spatial properties of events distinguished by temporal patterns (e.g., identify the location of a sensor prior to failure). In both cases I provided a controlled introduction to the visualization software interface, and allowed the participant to feel comfortable using the software before any analysis questions were presented.

a. Temporal Analysis

The initial user study employed a within-subjects experiment design and exposed participants to three different interactive visualization implementations. Because all three visualizations were implemented within the same environment, the interaction mechanisms were the same for each visualization. I focused this study on evaluating the effectiveness of this method for temporal analysis tasks. Temporal analysis tasks in this context require the user to identify a specific time that a temporal pattern occurs. For example, a user might be asked to identify the time that a failure occurred, or the time that a coordinated spatial activity began. Therefore, I ensured that participants understood how to recognize both individual and coordinated activities and sensor failures. These analysis tasks relate directly to real-world analysis tasks reported by experts in which temporal analysis is required to determine the local or global nature of a failure (i.e., a failure is local if no other failures occurred in the same time period).

Participants were asked to answer a series of questions using each visualization interface on different portions of the sample data. The order of visualizations used by each user was approximately balanced within the group in order to avoid the impact of learning effects on the experimental observations. The first visualization method consisted of an interactive bar graph and line graph depicting the aggregate number of location reports per period, as well as a state plot depicting red state for periods with no sensor activity and green state for periods with sensor activity (see Figure 30). The length of the time period could be varied interactively. This method was similar in capability to applications previously being used by the analysts. The second visualization method was a simplified method similar to the proposed method except that some of the primary glyph features were removed (see Figure 33). The final visualization method was the proposed method as described here.

During the study, participants were asked to use each of the provided visualization methods to answer a series of ten analytic questions about the data. Four of the questions required counting prominent data features, such as counting the number of sensors with no temporal threshold violations. The remaining six questions required the participant to locate and identify unique features, such as the largest gap between sensor reports. The order that the three different visualization methods were presented to each participant, as

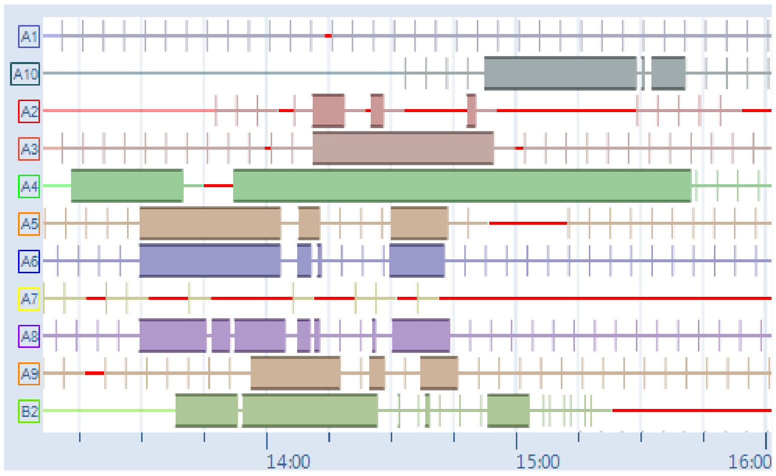


Fig. 33. Example of the simplified version of our method used for evaluative comparison. Threshold violations and reporting periods are still visible, however the triangular glyph is omitted.

well as the dataset used for each method was varied. The participants were instructed to indicate if they felt that the current visualization capabilities provided were not adequate to answer any given question, but were otherwise encouraged to answer each question to the best of their ability. We recorded the time and accuracy of each response. Because of the small number of participants, statistical evaluation of the results is omitted.

b. Spatial and Temporal Analysis

The second user study was designed to evaluate the benefit of this method for both temporal and spatial analysis tasks. It was focused primarily on detecting two anomalous behaviors common in spatial sensor systems: a failure to report when expected and reporting an erroneous location. In this study, I again asked participants to answer a series of ten analytic questions about a sample dataset, first using an interactive geospatial visualization coupled with a line graph showing the number of reports per sensor over time, and then using the geospatial visualization coupled with the proposed method. Both visualizations provided views with filtering, and linked selections and timelines, as shown in the figure on p. 106. Each session asked participants to perform tasks requiring spatial pattern finding, as well as geospatial referencing based on those findings. The last question of each session required participants to perform both spatial and temporal analysis (e.g., identify sensors exhibiting pattern X that were close together). There was an equal distribution of temporal and spatial tasks in each session.

4. Results

In this section I will present the results of this work using the user evaluation experiments described above, as well as expert opinion obtained from domain experts using this method to detect and analyze anomalies in the sensor network.

a. Scoring Method

In order to quantitatively evaluate the effectiveness of this method based on user evaluation studies, I will first define the scoring mechanisms for comparing time and accuracy between participants.

Time I recorded the time (in seconds) that each participant took to respond to each analysis question. I then computed the average response time per question for each visualization method, and the average response time for each task type (spatial or temporal).

I did consider whether to include the time scores for inaccurate responses, and decided that given the small number of participants in this study it was best to include all response times except those in which the user eventually indicated that he could not answer the question using the given visualization. Therefore this study evaluates the amount of time each user required to reach an answer using the current visualization technique independent of response accuracy.

Accuracy In order to measure a participant's response accuracy for each question, thresholds must be defined for both temporal and spatial tasks. That is, I will define how close an answer must be to be correct or receive partial credit.

To choose appropriate thresholds for these tasks, I considered that our domain experts reported that most events occurred over several days, however individual activities within the events usually spanned only a few hours. For both user studies, I presented participants with approximately two hours of data for each session. Therefore for temporal tasks, I regarded responses within one minute of the correct answer to be 100% accurate, while responses within five minutes received half credit and all other responses received no credit. The area of the simulation spanned a region of approximately 10 kilometers by 20 kilometers. For spatial tasks, responses within 100 meters received full credit while responses within 500 meters received half credit. These scoring thresholds were chosen based on typical analysis requirements indicated by domain experts.

For the final results of both groups, I first computed the accuracy scores for each participant's responses. I then computed the average accuracy by task type (spatial or temporal) for each different visualization technique used. Finally, I computed the average accuracy for each task type and each visualization method.

b. Experimental Evaluation

Both user studies were conducted using volunteer test participants, primarily graduate students in computer science. Because of light participation (four students in the first task, eight in the second), I cannot report useful confidence intervals; however I will discuss analysis of experimental results based on interpretation of large trends among the participants. I view these results as an initial indicator of the utility of this method.

c. Temporal Analysis Study

During the first study, all participants were able to answer every question using the proposed method, while at least one question was skipped for each of the other methods. As shown by the results in Table V, participants on average spent approximately 26% less time using the proposed method to answer the ten questions. Most importantly, I observed that when using this method, participants achieved noticeably greater accuracy than when using other techniques. In particular, accuracy in counting events and locating occurrences was doubled compared to the traditional visualization method and increased by approximately 30% compared to the simplified version of the proposed method.

Table V. Temporal Study Results

	Average Time (s)	Average Accuracy (%)
Traditional	76	0.38
Simplified	56	0.54
Proposed	56	0.74

d. Spatial and Temporal Analysis Study

For the second evaluation study, after computing average time and accuracy scores for temporal and spatial tasks, I observed an overall speed improvement of approximately 30% for temporal tasks and approximately 50% for spatial tasks. Accuracy for temporal tasks increased by 13%, while accuracy for spatial tasks increased by 10%. These results are shown in Table VI.

Based on these findings I believe that this method improves both speed and accuracy of analysts looking for temporal patterns or anomalies within spatial sensor performance data. In addition, I observed improved performance of spatial analysis tasks, when performed

Table VI. Spatial/Temporal Study Results

	Average Time (s)		Average Accuracy (%)	
	Temporal	Spatial	Temporal	Spatial
Traditional	62	136	0.69	0.74
Proposed	43	75	0.82	0.84

using interactive geospatial visualization complemented by the enhanced temporal pattern recognition provided by the proposed method.

e. Anomaly Detection

This approach enabled domain expert data analysts to detect several important anomalies within the sensor network. These can be divided into temporal and spatial/temporal anomalies. Confidentiality requirements prevent me from publishing examples using actual data collected by domain experts, however I have used representative datasets similar to the actual data to demonstrate these anomaly detection tasks.

Temporal Anomalies The primary temporal anomaly we sought to identify was a failure to report within the expected time for a particular sensor. This was often caused by a power failure, sensor failure, or loss of sensor connectivity (most were wireless sensors). Domain experts indicated that it was very important that faulty sensors were identified quickly to reduce errors in recorded data. Using this method, domain experts were able to quickly recognize and investigate these anomalies.

By using the proposed technique to analyze several datasets, experts began to recognize a common synchronized failure pattern of multiple sensors often occurring at the onset of a planned activity, as shown in Figure 34.After some investigation, it became clear that many players were in the habit of resetting their sensor systems

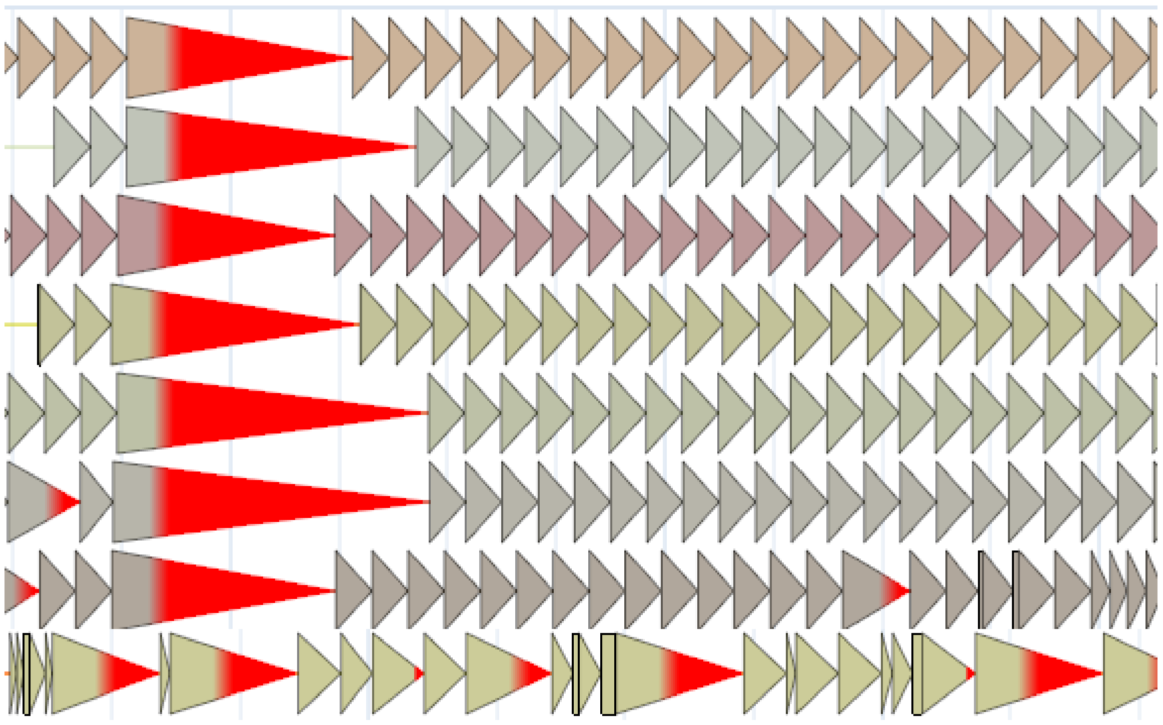


Fig. 34. Example showing an expected coordinated failure due to system maintenance on the first seven rows compared to erratic failures on the bottom row. This visualization allowed such faulty sensors to be identified and addressed quickly and easily.

before large coordinated activities. Subsequently, this failure pattern could be recognized immediately and therefore experts were able to focus attention on other anomalies.

Spatial Anomalies with Temporal Patterns Analysts were also able to identify several spatial anomalies in the sensor reports, meaning that sensors were reporting erroneous locations while moving along a route. As it turned out, the sensors being evaluated exhibited a unique temporal pattern in conjunction with the spatial reporting anomalies, and therefore those anomalies were able to be quickly detected using this technique, as shown in Figure 35.

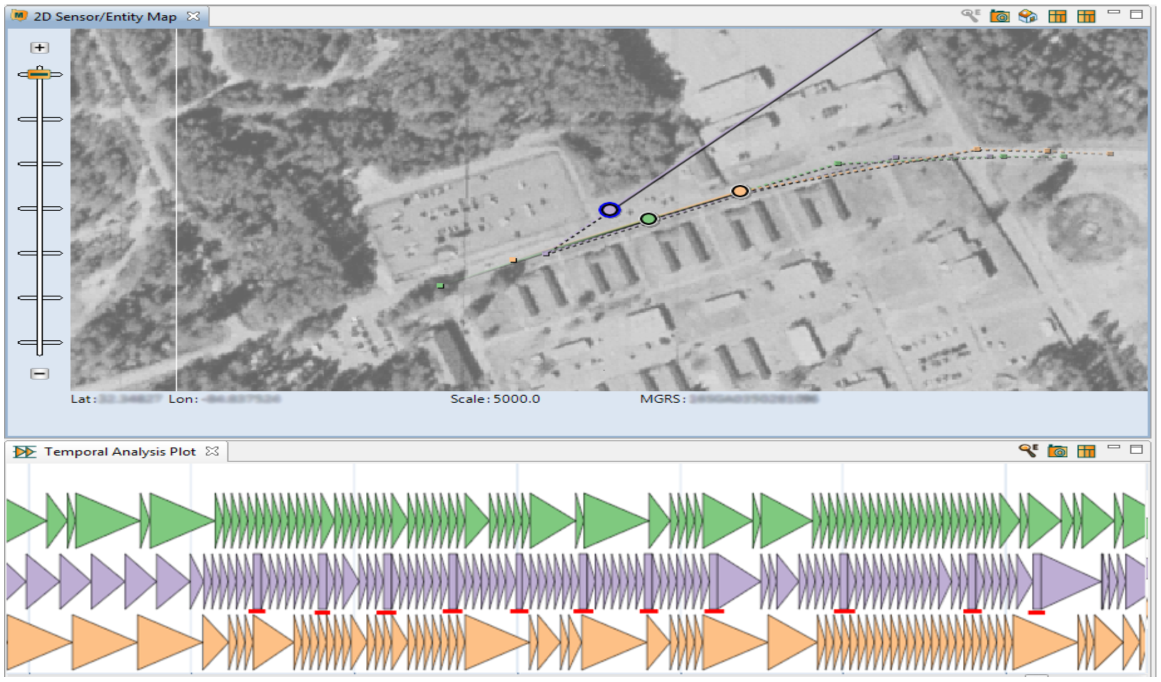


Fig. 35. Spatial anomaly (shown by diverging path) and temporal pattern (red marks have been added to highlight below second row). At this scale the rectangular areas clearly identify a change in the temporal pattern that accompanied the anomalous sensor reports. A total of 11 sensor anomalies occurred along the route. Each time this temporal pattern is observed, the system generates an erroneous position report at that sensor's last startup location. The use of our visual technique helped identify this repeated sensor failure, which turned out to be a design flaw in the sensor.

Based on this, geospatial correlations were recognized, indicating that certain terrain conditions caused a disruption in sensor connectivity (e.g., proximity to dense foliage). Therefore, an anomalous pattern that could previously only be detected by observing animated spatial visualization of sensor reports could now be discerned quickly over a larger span of time for a group of sensors simultaneously.

f. Expert Opinion

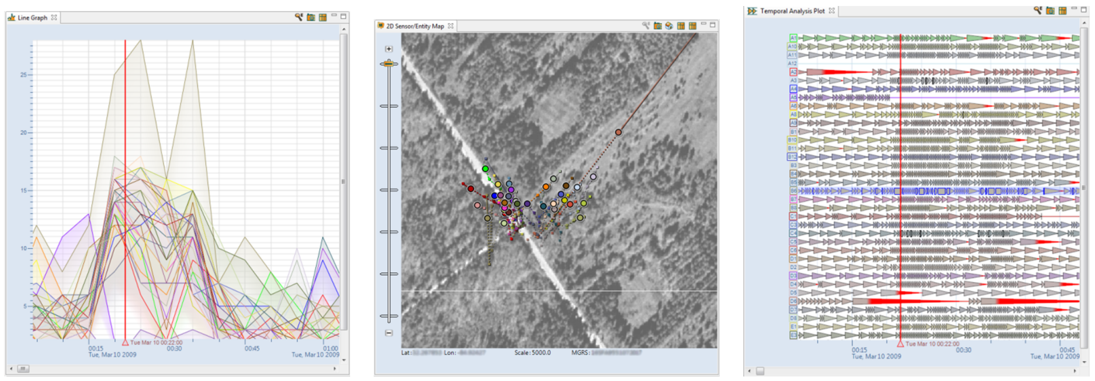
In addition to evaluation of empirical results of the user study, I also provided a fully-functional implementation of this visualization method to an expert team of analysts who were able to use the software to help analyze real datasets similar to the sample data described in this paper. There were a total of 12 expert evaluators. These evaluators were extremely familiar with the sensor equipment being monitored, and had years of experience using the traditional visualization methods to evaluate the equipment performance. Interviews conducted with the expert users after delivery, a short training session, and then unassisted use of the software revealed a high level of satisfaction in allowing the analysts to quickly perform visual inspection of large volumes of incoming data and rapidly identify and analyze potential anomalies. It should be noted that within the environment our method was deployed it was known as the Message Frequency visualization, due to the fact that the domain experts referred to each position report as a message. In particular, the expert users stated that:

- Our method allowed them to validate collected data in a few hours, a process that had taken several days using previous techniques.
- Our method provided a concise and accurate summary of a full day's worth of data in one image while visually highlighting all important data features.
- Our method provided effective output that could easily be integrated into presentation media to inform high-level decision makers.
- "We were able to use Message Frequency to estimate amount of data lost, by first identifying how much data was actually collected."
- "If we were to have had to calculate manually the times the system had communication breaks this process would take entirely too long...ProDV's Message Frequency

was the work horse that did all the 'data mining' for us and presented it in a easy to read visualization!"

g. Summary

To summarize the results of this work, I have evaluated the use of a novel visualization technique for temporal and spatial analysis of sensor data by implementing the method in a real-world analysis application, conducting a small user evaluation study, and reporting expert opinion from data analysts. While the sample sizes collected during the user study are not large enough to be conclusive, the collected data did indicate that our method was an improvement over traditional approaches being used by experts, shown again for comparison in Figure 36. Also, I will note several limitations with the proposed method.



(a) (b) (c)

Fig. 36. Example of sensor activity patterns in a line graph (a) and our method (c) shown next to an interactive map (b). For comparison, both (a) and (c) show the same range of data. Highlighting can be manipulated interactively to compare temporal patterns using (a), however sensor failures may not be as evident as they are in (c).

First, this technique has been applied to data that has a maximum rate of one event per second. Although our implementation should accommodate data with much higher rates, some of the visual analysis examples we have presented here may not be appropriate for such applications. Also, I have not attempted to address other common information visualization concerns at this point such as coloring to support color blind users or visual scaling for smaller or larger than normal (i.e., desktop) displays.

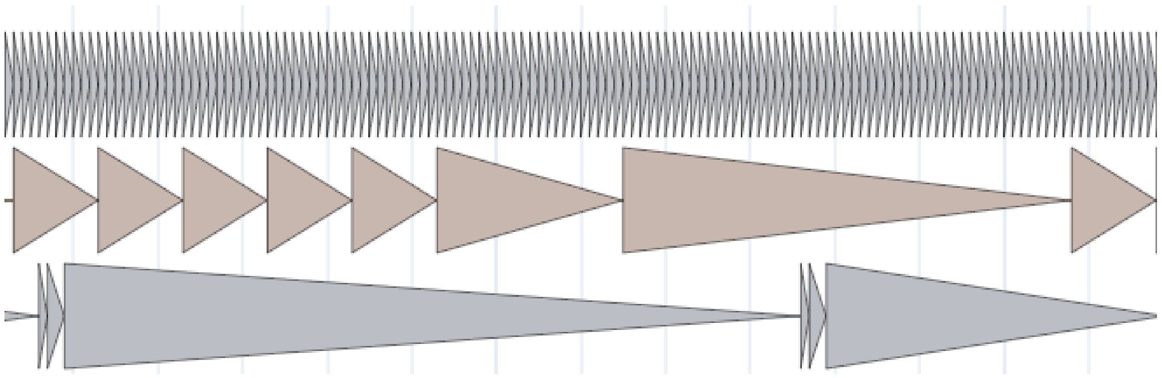


Fig. 37. Variation in temporal patterns produced by three different types of sensors attached to the same individual. In this case the first sensor was found to produce an excessive amount of redundant position data, even when not moving. Although all sensors were observed to generate position updates within the expected time, the final sensor type showed longer intermittent delays between updates and therefore the middle sensor was most appropriate for our users' task.

5. Future Work and Conclusion

There are several directions open for further work in this area. First, it would be useful to evaluate other analysis techniques that have been developed with domain experts, including comparative analysis of the performance of different sensor types attached to the same player. The goal of this analysis would be to identify which sensor platforms perform better under certain environmental conditions. For example, in Figure 37 we show a comparison of temporal sensor activity patterns generated by three different types of sensors all attached to the same vehicle. As can be seen, the choice of sensor will have a large impact on the

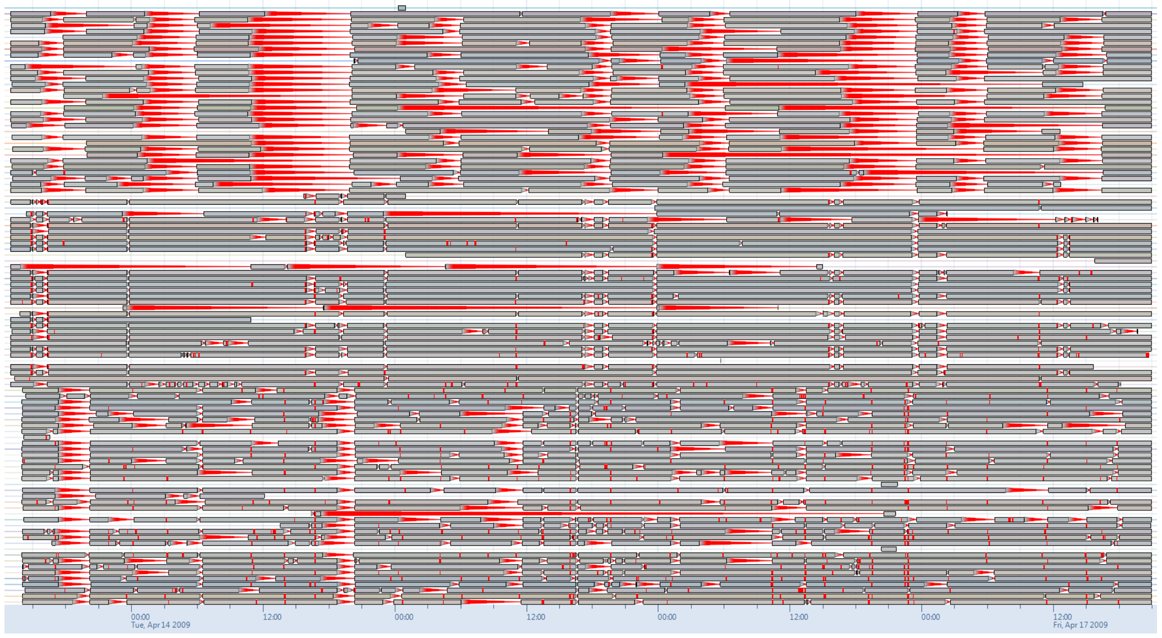


Fig. 38. Example showing five days of data from 102 sensors of three different types. Periodic maintenance patterns can be discerned and periods with high failure rates can be quickly identified. The three groups of different sensor types can be visually distinguished by their coordinated maintenance periods, and within these groups faulty sensors that do not match the group pattern can be easily recognized. While a large amount of data can be summarized in this visualization, at this scale it is difficult to discern finer details of the individual sensor activity patterns.

volume and accuracy of the sensor data.

It would also be useful to more formally evaluate the effectiveness of the analysis techniques we have presented at various scales. This will require identification of important temporal patterns at a larger scale than we have addressed in this user study. An example of this method applied to activity of over 100 sensors over five days is shown in Figure 38.

I have presented a simple yet effective method for interactive visual analysis of spatial sensor activity patterns. I have validated the use of this method in a real-world analysis application, and have presented the results of both expert opinion of the method and user evaluation studies comparing the analytic task performance of the proposed method to traditional techniques. Although the motivating problem and test data are quite specific, I believe that this work can be easily extended to provide interactive visual analysis of sensor or other temporal activity in other domains.

B. Situational Awareness

In this section I will discuss the second major domain application using the framework described previously. In this case the domain experts requested a computation and an interactive visualization of the results. Multiple commercial systems would be used to perform coordinated military activities that required a commander or other coordinator to direct multiple teams to achieve a goal. Each of the systems provided wireless nodes with advanced tracking and communications functions, and each was configured to record all of the message traffic that occurred during an exercise. The requested computation would determine the overall "correctness" of the situational awareness data (SA data) at each node in the system . I will discuss the details of the algorithm we proposed and implemented, the interactive visualization of the results, and my evaluation.

Situational awareness is made up of the information a person currently has available about his surroundings and relevant peers. Situational awareness data is often provided using electronic equipment designed to display real-time data from networks of spatial sensors. Because this information is usually delivered over a wireless network in a rapid decision-making environment, the latency of information delivery over the network could have a direct effect on human situational awareness and therefore decision-making. To be more specific, situational awareness data that is inaccurate can have a negative impact on decision-making involving coordinated activities if, for example, a user's perception of his peers' locations is incorrect. Since in this case the digital equipment is responsible for providing a large percentage of the visual data a user is assimilating at any one time to define his own situational awareness, these analysts desired a means to measure and compare the performance of different hardware implementations (i.e., from different vendors)).

In working with analysts attempting to evaluate and compare the performance of various SA systems, we identified a need for a set of custom visual metrics to support overview and detailed performance analysis of the data collected during test exercises. These exercises were designed to test actual users of the equipment in real-world scenarios involving coordinated activities so that the accuracy of different individuals' SA data can be compared and analyzed. For example, analysts would observe and compare the accuracy of a team leader's SA data and a team member's SA data during a convoy (a group of individuals or vehicles maneuvering along a similar path at the same time). We implemented software to compute the proposed metrics and provide interactive visualization to analyze the impact of network latency at specific times relating to decision-making efforts in the exercise. Using this method, we enabled analysts to quantify operational success by identifying any system that failed to deliver accurate SA information to the decision-maker at crucial times.

I will also discuss work with analysts who went on to use this technique in real data evaluation scenarios. These analysts provided significant input to the data collection process to ensure sufficient data was collected during the exercises in order to compute the proposed metrics. While I do not present user evaluation study for this technique I will present expert user testimony and several detailed analysis examples using these metrics in a custom interactive visualization environment.

For the purpose of this work, SA data consists of the most recently received position updates from all other peers (including the time the updates were generated). This type of data is referred to as SA Level 1 by Endsley [18]. In the analysis of SA data, analysts often refer to the most current SA data at a given time as the Common Operating Picture

(COP). The temporal metric was referred to by domain experts as "COP currency," since its purpose is to represent the accuracy of any individual's SA data at a given time. The proposed metric computation assigns a value on the range [0, 1] to each unique node (where 1 indicates an individual whose SA matches the current COP).

1. Proposed Method

This method focuses on defining metrics, temporal and spatial, that when presented in a visual framework will highlight failures in system performance. The temporal metric defines a measurement for each timestep (for each individual system), and the spatial metric defines a value for each grid location (for each timestep). Visualization of these metrics (in addition to a glyph-based two-dimensional map layer) was used to support analysis on each system's effectiveness to support human SA and decision-making. In this section I will discuss the proposed temporal and spatial metrics designed to support these analytic goals.

a. Temporal Metric

The temporal COP currency metric for an individual, X, is defined in (5.1), where n is the number of peers for X. Basically, this metric computes the percentage of peer locations accurate within some user-defined distance threshold.

$$CC_x = \frac{\Sigma_1^n \left\{ \begin{array}{c} 1 \text{ if } \left| p_i - p_{perceived\ by\ X} \right| < d \\ 0 \text{ otherwise} \end{array} \right\}}{n} \tag{5.1}$$

For a given individual, we calculate the COP currency as the fraction of peers for which the perceived position is current. A position is considered current if the actual position is less than some distance *d* (five meters in our exercises) from the perceived position (i.e., the most recent position update received). Analysts requested to see COP currency values at a 1 second resolution, therefore this implementation computes this metric for each individual for each one-second period of the exercise period.

In order to compute the proposed temporal metric, both send and receive times of every SA message for every individual in the exercise are required. Also, for the purpose of these exercises, the analysts directed us to use the send times of SA messages to define the current COP. We designed the system, however, to allow a separate source of location data to be optionally specified to define the "ground truth" COP. The receive times of SA messages are used to determine the perception of peer locations by any individual since that perception is defined by the most recently received SA messages by the individual from his peers.

See Figure 39 for an example of the COP currency metric shown along with a linked geospatial visualization depicting COP locations for each individual as well as perceived peer locations for the selected individual (orange).

b. Spatial Metric

In order to capture spatial COP currency failure patterns, we also defined a spatial metric that accounts for the existence of failures within discrete areas of the exercise location. First we initialize a two-dimensional grid that covers the geospatial exercise area as defined by the minimum and maximum positions reported by any unit in the dataset. The dimensions of the grid can be set manually or set automatically based on desired spatial resolution (for the exercise areas we observed, a manual setting of 1024 x 1024 for the grid size was used). For each second in the exercise, as we compute each individual's COP currency value based on SA data, we mark the positions of the peer and the individual for each threshold violation (when distance between local SA and COP is greater than *d*). We mark the location of the

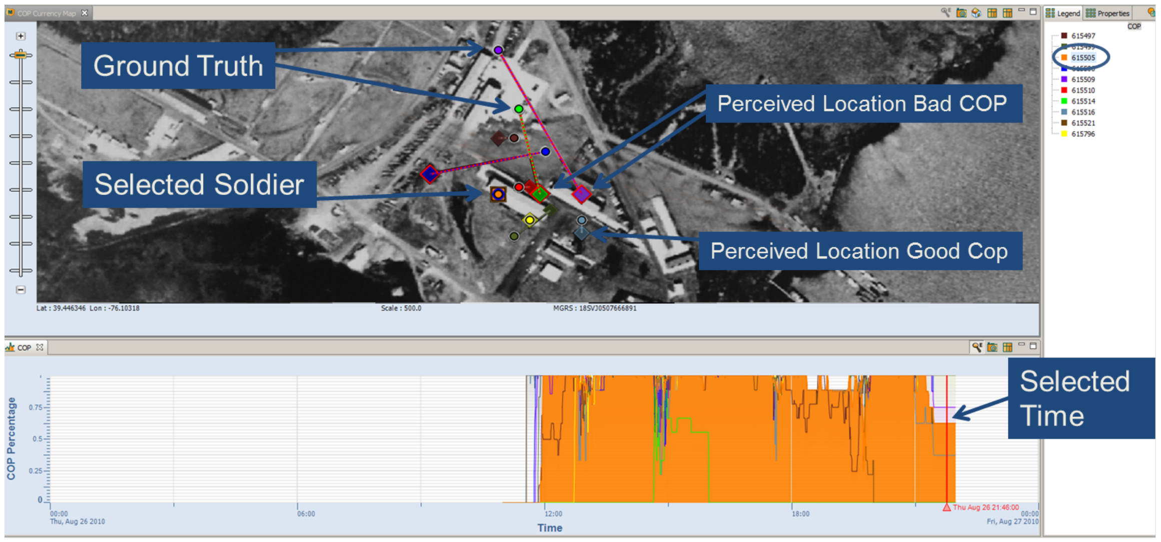


Fig. 39. The COP currency temporal metric is shown in this example in the line graph at the bottom of the coordinated view. COP currency value is low for the orange individual at the selected time (vertical red line on line graph) since the locations for a majority of peers is not accurate (highlighted with red lines and diamonds on map). The dotted red lines on the map depict the difference between COP position and perceived position of individuals. The common control shown (legend and filter on right, timeline on bottom) are provided by the visualization environment we have used, ProDV.

peer as *sender violations* and the location of the individual as *receiver violations* since the peer and individual constitute the sender and receiver of SA data, respectively. Therefore, our spatial metric at each grid cell in the exercise area will mark each grid location in which send or receive failures have occurred. The spatial algorithm is aggregated over the entire exercise and can then be displayed as a colored overlay as shown in Figure 40. This technique allowed the analysts to observe and investigate any spatial failure patterns of a particular system.

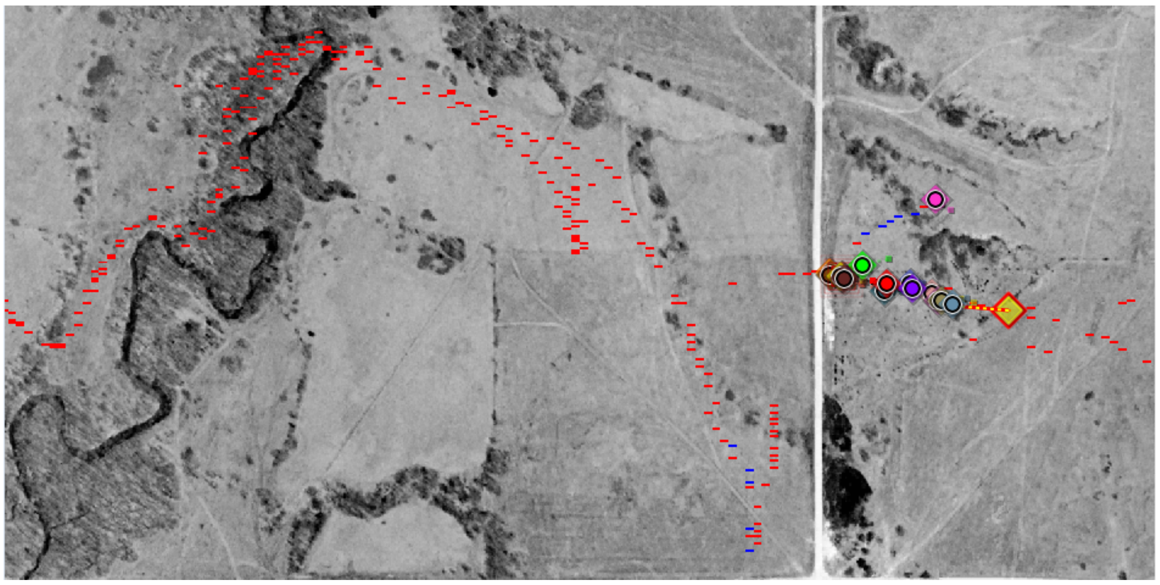


Fig. 40. The geospatial display at the top of this coordinated view shows the violation areas identified by the spatial metric. In this example, areas that contain both send and receive violations are highlighted in red. A potential negative relationship between proximity to foliage and COP currency can be observed, although the SA system being evaluated in this example seems to have high failure rates in some open areas as well. The vertical white line on the map is a road.

2. Implementation

In this section I will discuss the interactive software implementation of this method that was delivered to analysts at the conclusion of this work and used for evaluation in a real-world data analysis setting. This implementation again made use of the existing data processing and visualization framework, ProDV [55], that included capabilities for reducing and visualizing SA data produced by the equipment employed during these exercises. The implementation of the metric computations were encapsulated within a new processing module that provides data to a geospatial visualization component. We also implemented a visualization of threshold violations as well as the spatial metric by defining a new layer to the existing visualization component. Line graphs and bar charts were used to display the temporal metric in an interactive graph linked via selection to the map and other visual controls (timeline, legend, filter).

a. Data Processing

The data processing logic for our implementation consisted of a two-pass algorithm. Because the raw input data was harvested in batches from the equipment and processed directly by ProDV after the exercise, there was no guaranteed ordering of the input data. Therefore the first pass of our algorithm simply sorted SA message send and receive records into bins (one for each hour of the exercise, in our case). After the first processing pass, we are left with reduced (only time and position information) and ordered SA data for each individual. The second pass iterates through the exercise time period (defined by the minimum and maximum times observed in the first pass) at a one second step size and computes both temporal and spatial metrics.

Figure 41 depicts the reduced data structure for individual X produced by the first pass of the data processing algorithm, composed of the series of SA messages sent by X (COP_X), and the receive times of those messages by each peer (SA_Y and SA_Z). The vertical red line in the figure illustrates how we iterate in the second pass through the reduced data structure to compute the temporal and spatial metrics at a one second interval. The temporal metrics are stored in bins (one per hour) similar to the first pass data, and the spatial data is stored in a two-dimensional array of integer values.

b. Visualization

The ProDV visualization environment provided basic temporal and spatial visualization capabilities using line graphs and geospatial displays. In the following sections I will discuss the temporal and spatial visualization implementations.

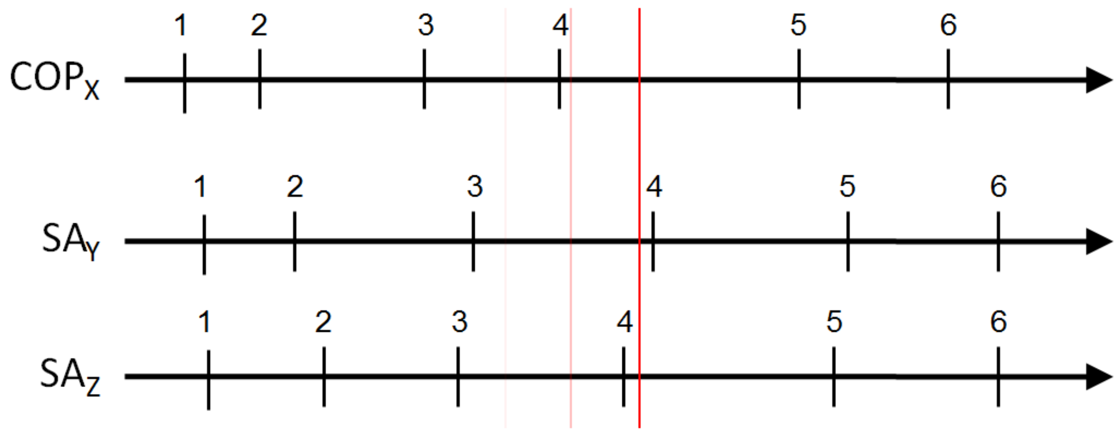


Fig. 41. Data structure for computing COP currency metrics. In this example individual *X* sends out six SA messages (shown in first row), which are received by peers *Y* and *Z* with varying latency (shown in bottom two rows). At each timestep (one second), for each individual we retrieve the COP position (defined by sent SA data from each individual) and perceived position for each peer (defined by locally received SA data), and compute the metric value as shown in (5.1).

Temporal Metric Visualization As discussed, the temporal metric value is computed for each individual at 1 second intervals within the exercise time frame. The value for each individual is displayed in the interactive line graph as shown in Figure 42. The interactive graph is linked to legend selection and filter operations. Likewise selection within the line graph is linked to the map visualization, so that individuals can be selected for their temporal or spatial attributes, and can then be highlighted in other visualizations.

Threshold Violation Visualization We implemented a spatial visualization feature to depict COP currency threshold violations. The existing ProDV geospatial visualization depicted individual locations using circular glyphs that could be augmented with trails to show the recent path of each individual. Because it would certainly clutter

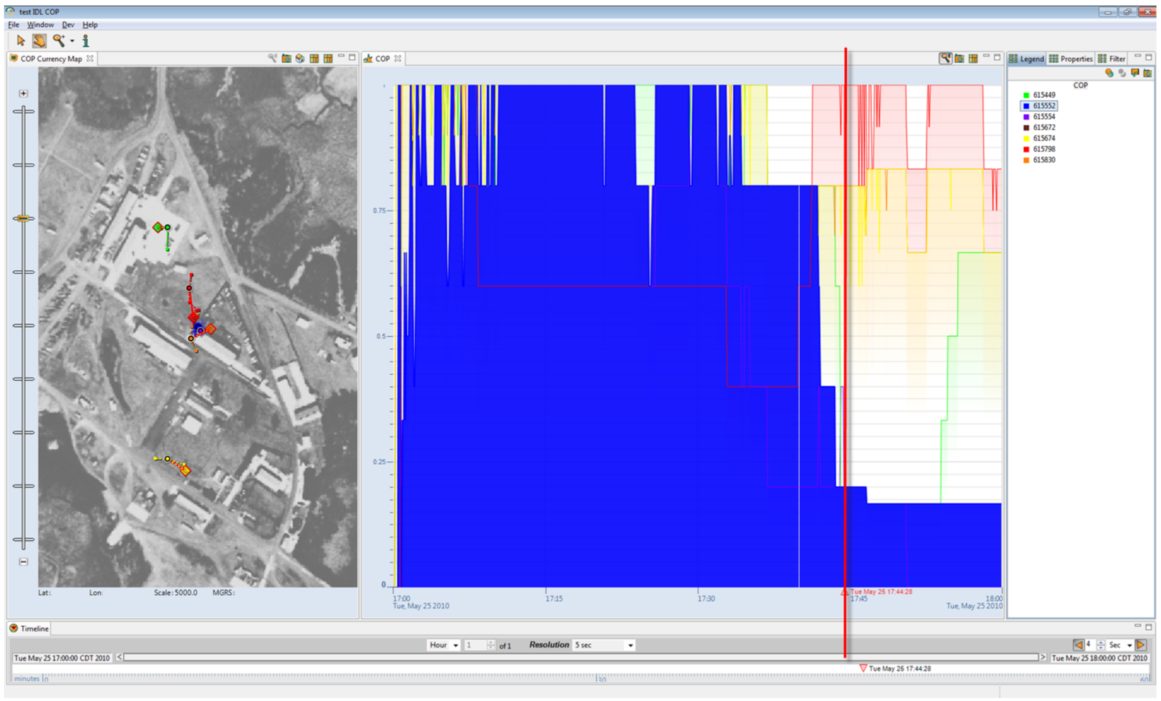


Fig. 42. Interactive line graph showing temporal COP currency metric. The blue individual is selected, prompting the color highlighting shown in the line graph, and the geospatial view has been automatically centered on the blue individual's location. At this time there are multiple peer COP currency violations as shown by the low COP currency value at the selected time (vertical red line).

the visualization to attempt to display the SA data (perceived peer positions) for all individuals, we chose to show the additional SA data only for one selected individual at a time.

As shown in Figure 43, when an entity is selected the perceived locations for his peers are shown as transparent diamonds connected by transparent dotted lines to the actual locations designated by circles. If the perceived location is outside of the defined spatial threshold, the diamond and connecting line will be outlined in red to highlight the violation.

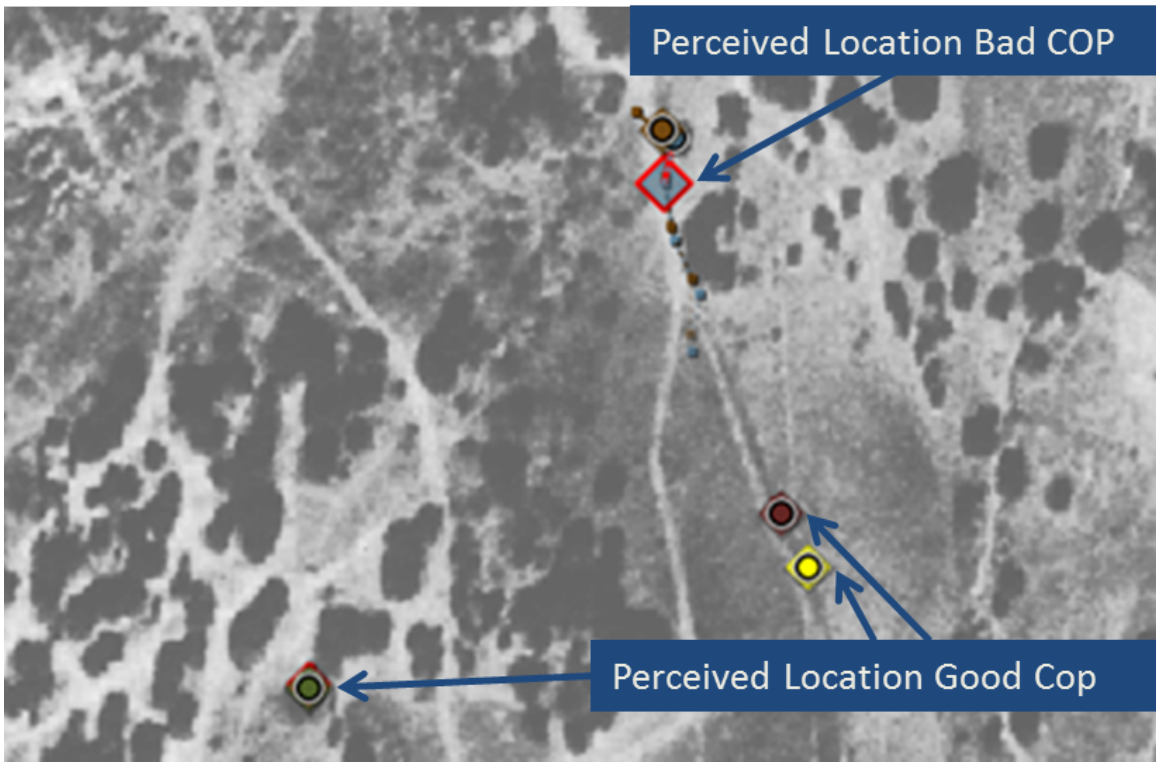
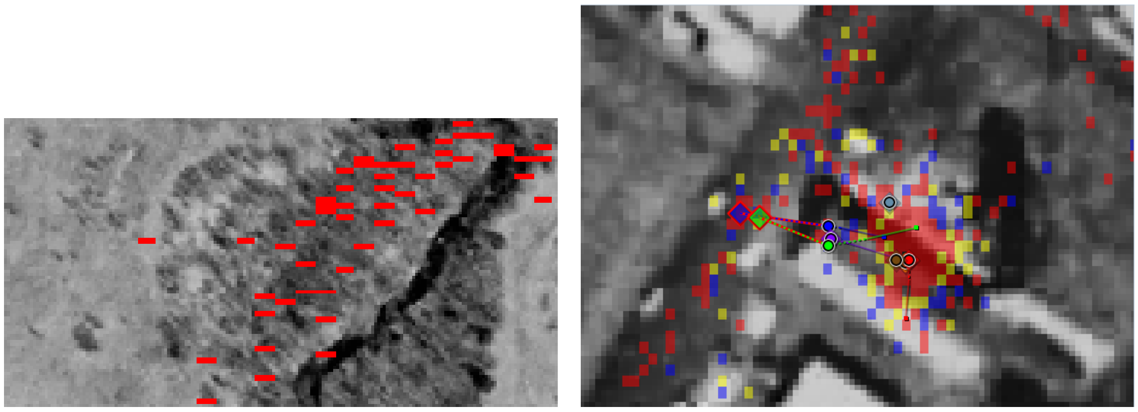


Fig. 43. This COP currency visualization example shows a group of individuals moving along a dirt road (trail points indicate recent spatial history). The selected individual is not shown, and all peers shown are current except one (light blue), whose perceived position (red diamond highlight) is greater than 5m from the COP position.

Spatial Metric Visualization The spatial metric is displayed as a transparent layer on the geospatial visualization. The two-dimensional array of values are loaded as a texture in OpenGL and sampled using a custom fragment shader implemented using GLSL. For example, we can highlight cells red when there are both send and receive threshold violations and blue or yellow when there are only send or receive violations, respectively. Examples are shown in Figure 44, in which we can observe areas where failures were more likely to occur over the course of the exercise. Alternatively, we can adjust the opacity or color intensity of the highlights to indicate the number of violations recorded in the area, however it is well-known that the human eye is not suited to discern small variations in these visual effects, which are already plotted over existing spatial data in our visualization. Therefore we simply allow a violation count threshold to be adjusted interactively so that analysts can identify areas that experiences higher numbers of violations during the exercise.



(a) (b)

Fig. 44. COP currency spatial metric visualization examples in which areas with both sender and receiver COP currency threshold violations are highlighted in red. In (a) we can see that most threshold violations occur within the wooded area, as opposed to the open area, both heavily trafficked over the course of the exercise. In (b), we also marked areas with send only or receive only failures with blue and yellow highlights, respectively. The map in (b) shows a diagonal road alongside an elongated building, which appears to have had a negative impact on COP currency of all units in the area.

c. Interaction

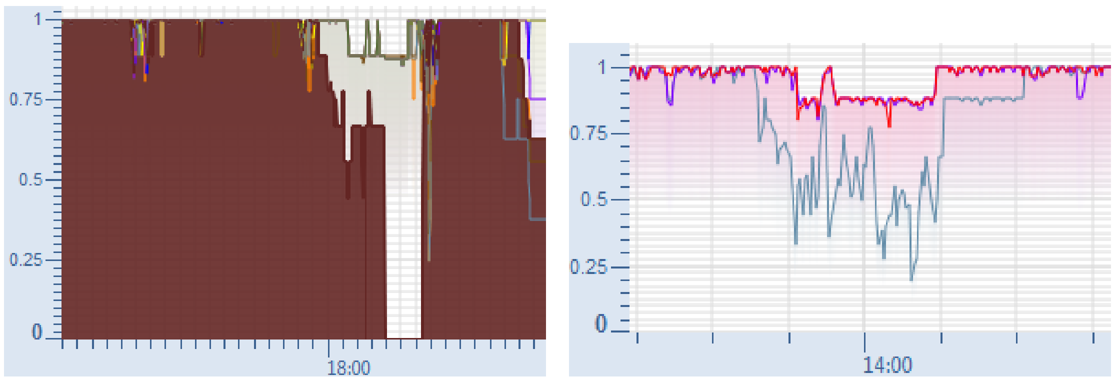
The ProDV visualization environment provides select, zoom, pan, and query tools for each visualization. These basic interaction features provide most capabilities required, and selection between visualizations and other common controls such as the legend are automatically synchronized. A timeline control provides the ability to navigate the temporal dimension of data, while the map interactions provide intuitive spatial navigation capabilities. With these existing interactions in place, experts could easily examine the temporal metric values for a given individual, or for all individuals at a given time. They could also easily examine the location of individuals in relation to metric values and identify individuals in close proximity and compare temporal metric values. In the implementation of this work we did not add any new interaction mechanisms, but we did link the spatial visualization of threshold violations and perceived peer selection to the existing selection framework and customized some visual queries used in the spatial visualization.

3. Analysis

Use of actual exercise data may not be published, so instead I have provided here synthetically generated data that has similar characteristics to that seen in real exercises. I will describe three analysis tasks that closely mimic those of actual analysts. The analysis tasks can be classified as either temporal analysis activities, spatial analysis activities, or system performance analysis activities.

a. Temporal Analysis

The primary temporal analysis task identified by domain experts was detailed failure analysis of COP currency threshold violations. When COP currency decreases for an individual, analysts want to know if other individuals in the exercise experience a synchronized decrease as well. If so, then there is likely a global system performance issue that affects SA performance for all individuals (e.g., a network failure). Otherwise it is likely that there is a local equipment failure for the individual that should be investigated. Figure 45 shows an example that compares a local failure pattern to a global failure pattern.



(a) (b)

Fig. 45. In this example we see a local decrease in COP currency (selected brown unit is highlighted for readability) in (a) in which only one individual system seems to experience a dramatic failure to maintain accurate SA data. In (b), however, we observe a decrease in COP currency of several systems during a synchronized time period. This observation could indicate a network failure or some other external failure that affects all systems.

b. Spatial Analysis

In addition to temporal analysis of threshold violations, experts were also able to conduct analysis of spatial failure patterns. First, building on the temporal analysis discussed previously, when global failures occur the individuals affected by the failure can be selected and any pattern in the location of the individuals can be quickly seen. For example, individuals who are in a heavily-wooded area or an area obscured by terrain to are expected to experience network latency sufficient to reduce COP currency, as shown in Figure 44. However an individual experiencing reduced COP currency while in in open terrain and near other units with stable COP currency is likely due to an equipment failure or some other local anomaly.

c. System Performance Analysis

Other more complex analysis tasks focus on evaluating the performance of the SA system in its primary mission to deliver timely and accurate SA data to the user. Two tasks in particular that were identified by expert users were the evaluation of a group leader's COP currency during various scenarios, and the performance of the SA system during convoy operations.

Group Leader SA During coordinated activities, the accuracy of the SA data provided to the team leader is critical, and therefore analysts desired to evaluate this directly. Using the software implementation of the proposed technique, the experts were able to select the identified team leader by ID in the legend, and using the timeline select the approximate time of the specific activity to be analyzed. In most cases hard-copy spreadsheets were used to record event data during exercises. Users could choose to import this data into a synchronized interactive timeline, or just use the hard copies (in most cases the hard copies were used). Once the appropriate time is selected the analyst can quickly identify any unexpected variations in COP currency using the line graph. By animating the motion of the team members on the map visualization, the analyst is also able to identify potential geospatial features that impact team leader COP currency during the coordinated activity.

In the example shown in Figure 46, the group leader assumes a static position as team members move to an observation point. We can observe different COP currency profiles for each different member of the group. In this exercise, the group leader's COP currency values are quite different from those of the other group members.

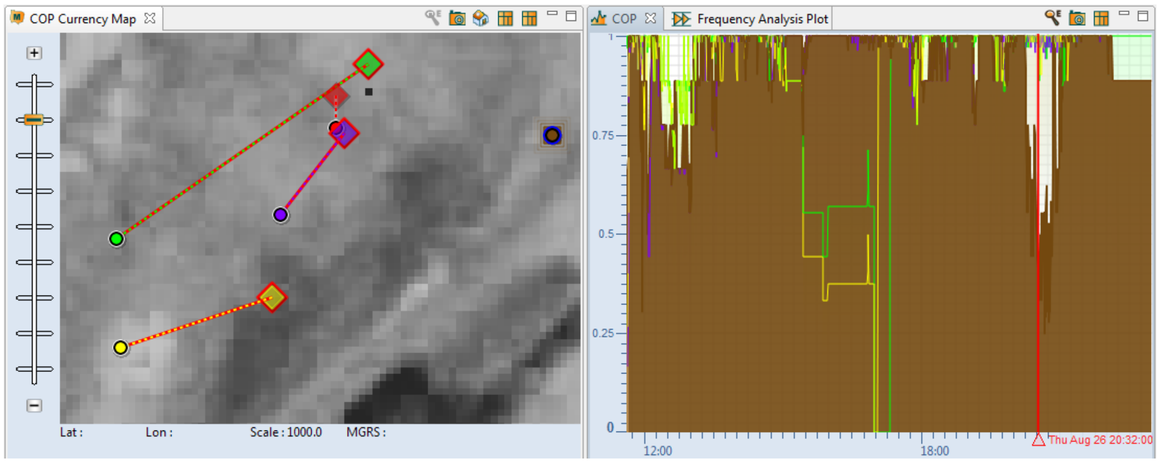


Fig. 46. In this example we observe a group of five individuals involved in a coordinated activity. The group leader (brown) remains stationary while the other group members gather at a designated location. We can see that COP currency values fluctuate for all group members at different points during the exercise. In this scenario the group leader did in fact not have accurate SA data at a critical decision point marked by a vertical red line near the right edge of the line graph.

Convoy The next specific analysis task identified by domain experts was the analysis of COP currency during high-speed convoy operations. In particular, analysts wanted to be able to observe variation in COP currency of both moving and non-moving systems when a group of individuals were involved in a high-speed coordinated movement. To do this, analysts would perform temporal sweep of the COP currency values of the individuals involved in the motion activity, since individuals in a similar location executing a synchronized motion are expected to experience similar variation in COP currency. We also include the custom glyph visualization previously developed to aid in the analysis of sensor reporting activity, shown in Figure 47, in order to aid in the identification of these coordinated periods of movement [54]. Because the sensors used in these exercises exhibit increased reporting activity when moving, the visualization technique shown allows coordinated periods of motion to be quickly identified and enabled quick multiple-selection of individuals linked to the other visualizations.

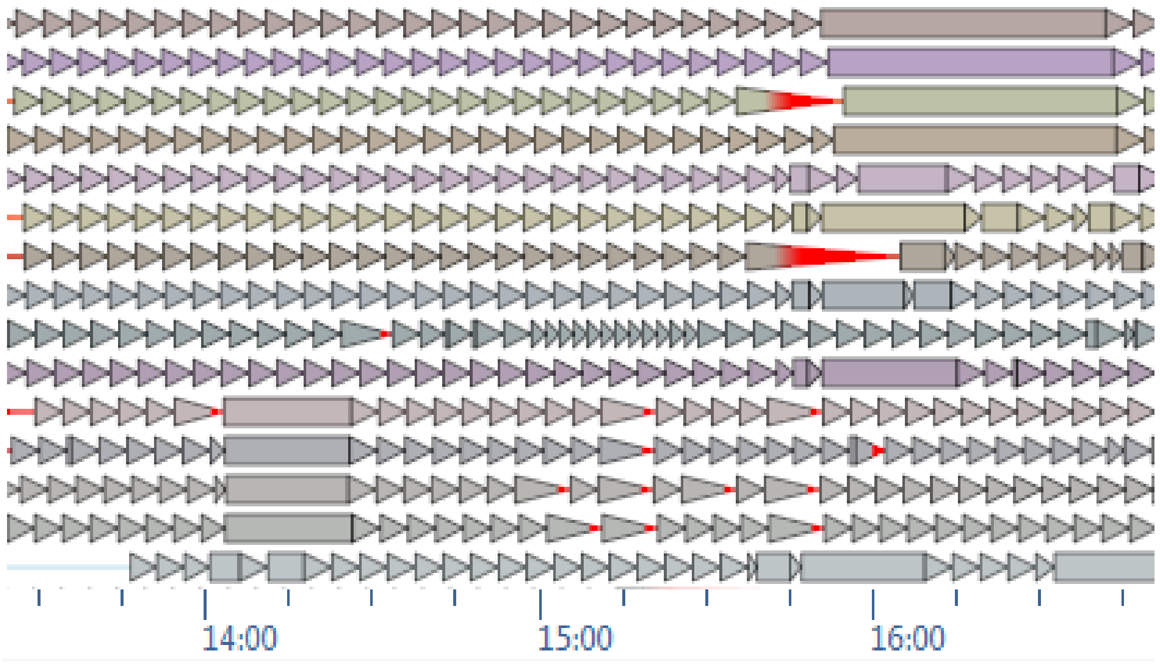
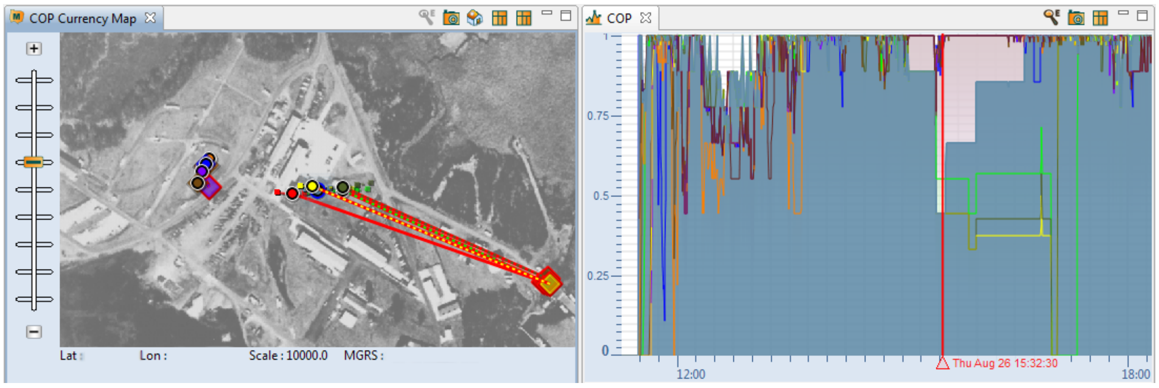
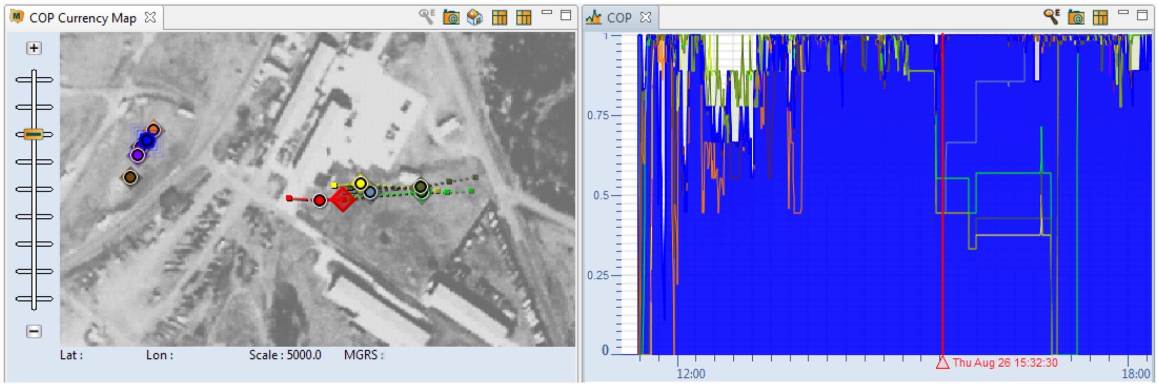


Fig. 47. Custom visualization of spatial sensor reporting activity aids in the identification of sensors in motion. Triangles represent periods of time in between position update messages. Triangles not visible at the current scale are aggregated into rectangular regions. Because the sensor activity glyphs are arranged in rows, coordinated activities of multiple sensors can be quickly identified. In this example, two coordinated movements are visible: one around 14:00 by at least four sensors, and another around 16:00 involving several more individuals.

Figure 48 shows a comparison of two different perspectives at the same moment during a convoy operation involving five individuals. The first figure shows the perspective of a member of the convoy operation who is in motion. As can be seen in the line graph and geospatial display, the selected individual's SA data is inaccurate for every peer in the convoy group.



(a)



(b)

Fig. 48. In this example, an observation group is stationed to the left while a group of four individuals move quickly in a coordinated motion. We can observe the SA perspective of the light blue individual in (a) and notice that this individual, participating in the convoy, has a low COP currency value since SA data for all of his moving peers is inaccurate. In (b) we can see that the SA perspective of the stationary observer in blue is accurate for three of the moving peers, and likewise COP currency value is high. This example illustrates the COP currency temporal metric being used to identify a potential system weakness delivering accurate SA data during convoy operations.

Conversely, the perspective of a stationary observer in an open area is shown in the second figure. Not surprisingly, the SA data for the stationary observer is much more accurate. This example illustrates the power of our temporal metric for quantitative evaluation of SA data accuracy in the analysis of the performance of a particular SA system in varying conditions.

4. Results

The publishable results of this work are comprised of expert opinions as reported by analysts trained to use this technique implemented using ProDV in real-world analysis scenarios. As mentioned previously, examples of specific analysis results and system comparisons were not available for publication, however I believe the generic analysis examples discussed previously as well as the expert user testimony serve to support the proposed use of these metrics in an interactive visualization environment to support human analysis.

After use of this visual analysis software, the expert users reported that:

- “This method allowed evaluators to analyze COP currency visually, and this not only simplified data analysis but simplified our whole process.”
- “The COP currency visualization allows the analyst to identify critical events and determine quickly how the system under test was performing.”
- “Post-test review of the visualization gave the analyst the capability to focus on both good and bad system performance, and link the critical variables that led to that performance.”
- “Looking at COP currency metrics in a tabular form does not tell the analyst anything about why those numbers exist; looking at COP currency in a dynamic visualization

gives operational relevance to the numbers."

- "I immediately saw the benefits of [this technique] and since the first day of our classroom instruction at the software lab I have been very interested."

5. Conclusion

I have proposed a temporal metric to quantify COP currency of individuals in an exercise, as well as a spatial metric to identify spatial failure patterns. I have discussed how these metrics have been implemented and delivered to expert analysts in an interactive visualization environment. The successful use of the proposed methods demonstrate their effectiveness in improving a human analyst's ability to quickly and effectively understand and evaluate the outcome of exercises designed to evaluate the performance of SA systems.

CHAPTER VI

CONCLUSIONS

A. Statement of Contributions

In this dissertation I have presented novel contributions to the field of computer graphics and visualization, both in terms of the models used to design interactive visualization systems for large numeric datasets and the application of design principles such as data caching and user interaction modes to specific visual analysis challenges. First, I have introduced a modified model of the information visualization reference model that enables interactive visualization techniques to be applied to large numeric datasets by first reducing and caching intermediate data. I have demonstrated the success of several different interactive visualization applications using a software framework based on this model. Using this technique, developers of interactive visualization software can design custom visualization techniques and apply them to very large collections of data. This capability is important to aid data analysts who are critical components of important decision-making and scientific discovery processes in the visual analysis of the massive volumes of data that are available today.

Second, I have presented a technique for efficiently implementing view-prioritized asynchronous retrieval of high-resolution surface data based on an adaptation of the graphics hardware pipeline, and I have demonstrated how this technique can be applied to interactive terrain visualization using high-resolution satellite data. I have shown that the software implementation developed using this technique is able to provide interactive rendering performance that is independent of the size of the remote terrain database, the latency of retrieval of the imagery data, or the resolution of imagery data used. This technique will enable developers of interactive visualization software to design and implement interactive visualizations of datasets whose size and retrieval cost are much higher than ever before. This is important to researchers and users of visualization software in many scientific disciplines that require visual exploration of large amounts of data collected at macro- (astronomical data) or micro- (genomic or particle physics data) scales. Because many scientists are required to visually analyze these large collections, it is not practical to provide the full dataset to each user. This technique allows complex 3D interactive visualization techniques to still be applied to these datasets to aid in this analysis, regardless of the performance or capabilities of the remote database system.

Lastly, I have demonstrated the successful development and application of novel visualization techniques designed to aid in two distinct visual analysis tasks. First, I have presented a novel technique for the visualization of temporal patterns of sensor activity. This technique was developed using the software framework I have proposed to ensure interactive performance when working with large datasets, and it enabled real-world analysts to quickly and accurately perform several important data analysis tasks. I also presented a novel technique for quantitatively measuring the accuracy of digital situational awareness data and visualizing this information in temporal and spatial contexts. This capability has also been applied to very large datasets and has allowed analysts to inspect the performance of a digital situational awareness system in detail and compare the performance of different systems in experimental conditions.

B. Future Work

Several directions for future work based on the techniques presented here are worth exploring. A shortcoming of the work presented here is the statistically significant evaluation of the advantages the proposed visualization system framework. As discussed in Chapter III, efforts to collect significant amounts of user data during software training sessions proved to be too distracting to most users and therefore not enough user performance data could be collected. As a future direction for this work it would be helpful to integrate a near-ubiquitous user evaluation mechanism whereby a user working with an interactive visualization of a sample dataset during a training session could be automatically prompted with small timed analysis tasks. This effort would improve and streamline the performance evaluation of individual visualization techniques, and therefore contribute to the broader evaluation of the improvement offered by this approach as a whole. Another limitation of this technique is that it assumes that the data reduction strategy specified by the user is sufficient to reduce the numeric dataset into manageable block sizes. This approach works very well with evenly-distributed numeric data, as I have shown in several examples. However, very dense, unevenly distributed data can pose a challenge that will require additional considerations to ensure interactive visualization performance is possible. Possible approaches that might help address this limitation would be dynamic segmentation of intermediate data or use of reduction methods that recurse adaptively.

The technique presented here for the view-prioritized asynchronous retrieval of remote data could also be applied to visual analysis of other spatial data (not imagery, e.g., results of a high-resolution propagation or dispersion simulation) or any object with complex occlusion and high-resolution surface data. In this case the primary considerations are the texture formats required to accurately store the geometry and imagery data (e.g., 32-bit integer vs. 64-bit floating point data) and the number of other texture elements required by other components of the visualization. Also, the use of mipmap memory areas for index storage can be improved by computing the maximum mipmap layer (smallest number of pixels) that can be used and still provide adequate sampling resolution. If the maximum required level is greater than the number of levels required to store the index, then intermediate mipmap levels can be populated with actual texture mipmaps, which can be used to improve rendering quality. Lastly, the current method of sampling data indices into an offscreen buffer introduces a noticeable pop when moving quickly within an interactive 3D environment. I have minimized this effect in the implementation by only sampling the indices when the user stops interacting (i.e., releases the mouse button or key), or after a specified delay is reached (five seconds in this case). However, it would be better to improve this implementation by either pre-rendering a visual frame before sampling offscreen or adaptively adjusting the view parameters to minimize distortion. While the current implementation might support the latter alternative, the Java-based OpenGL implementation I have used for this work does not provide adequate control over the scheduling of rendering calls, making it difficult to control buffer-swap operations necessary to ensure smooth visual quality in cases when the data indices need to be sampled while the user's 3D viewpoint is being rapidly altered.

For the glyph-based visualization technique for temporal data I have presented, this work could be improved by executing additional user testing. For this work the user evaluation study took volunteer participants at least 90 minutes to complete. The length of the study and the equipment required to conduct and monitor the study made it difficult to recruit a larger pool of participants. It would be advantageous to design a web-based user evaluation study that could be conducted using any computer with a standard web browser and Internet connection. The addition of more participants over the web would increase the statistical significance of the user performance results presented here. Also the aggregation technique described (i.e., combining sub-pixel triangles into quadrilateral regions) could be taken one step further to allow sensor groups to be aggregated. This would be particularly useful in cases where analysts need to inspect the performance of a group of sensors acting as a unit (i.e., the system is considered to be performing normally as long as one sensor in the group is operational).

With respect to the technique I have presented to qualitatively measure and visualize the accuracy of situational awareness data, there are several directions for future study.

In particular, some analysts mentioned that it would be useful to interactively designate outlier data to be removed from the metric computation. The implementation of this method computes the metric in two passes, only the second pass of the metric computation would need to be executed again in this case. For this technique it would also be useful to explore additional visual aggregation of data so that the accuracy of situational awareness data can be measured and compared between groups of systems. Appropriate spatial filtering tools can be integrated to allow this technique to be extended to aid in the discovery of spatial patterns in the performance of different sensor groups, which may be caused by spatial or environmental effects.

REFERENCES

[1] W. Aigner, S. Miksch, W. Müller, H. Schumann, and C. Tominski. Visualizing time-oriented data: A systematic view. *Computer Graphics*, 31:401–409, 2007.

[2] P. Bak, F. Mansmann, H. Janetzko, and D. A. Keim. Spatiotemporal analysis of sensor logs using growth ring maps. *IEEE Transactions on Visualization and Computer Graphics*, 15(6):913–920, 2009.

[3] X. Bao and R. Pajarola. LOD-based clustering techniques for efficient large-scale terrain storage and visualization. In *Proc. of the SPIE Conference on Visualization and Data Analysis*, pages 225–235, 2003.

[4] S. Basu and J. Snoeyink. Terrain representation using right-triangulated irregular networks. In *Proc. of the Canadian Conference on Computational Geometry (CCCG)*, pages 133–136, 2007.

[5] A. Bayoumi, M. Chu, Y. Hanafy, P. Harrell, and G. Refai-Ahmed. Scientific and engineering computing using ATI stream technology. *IEEE Design and Test*, 11:92–97, 2009.

[6] Berkeley University of California. Prefuse. http://prefuse.org, Accessed on November 1, 2011.

[7] I. Buck, T. Foley, D. Horn, J. Sugerman, K. Fatahalian, M. Houston, and P. Hanrahan. Brook for GPUs: Stream computing on graphics hardware. *ACM Transactions on Graphics*, 23:777–786, 2004.

[8] S. K. Card, J. D. Mackinlay, and B. Shneiderman. *Using Vision to Think: Readings in Information Visualization*. Morgan Kaufmann Publishers, San Francisco, CA, 1999.

[9] J. V. Carlis and J. A. Konstan. Interactive visualization of serial periodic data. In *Proc. of the ACM Symposium on User Interface Software and Technology (UIST)*, pages 29–38, 1998.

[10] E. H. Chi. A taxonomy of visualization techniques using the data state reference model. In *Proc. of the IEEE Symposium on Information Vizualization (INFOVIS)*, pages 69–75, 2000.

[11] W. Chung, H. Chen, L. G. Chaboya, C. D. O'Toole, and H. Atabakhsh. Evaluating event visualization: A usability study of COPLINK spatio-temporal visualizer. *International Journal of Human-Computer Studies*, 62:127–157, 2005.

[12] A. Clauset, M. E. J. Newman, and C. Moore. Finding community structure in very large networks. *Physical Review E*, 70(6), 2004.

[13] K. Cook and J. Thomas. *Illuminating the Path: The Research and Development Agenda for Visual Analytics*. IEEE CS Press, 2005.

[14] C. Dick, J. Krüger, and R. Westermann. GPU-aware hybrid terrain rendering. In *Proc. of IADIS Computer Graphics, Visualization, Computer Vision and Image Processing*, pages 3–10, 2010.

[15] M. Duchaineau, M. Wolinsky, D. E. Sigeti, M. C. Miller, C. Aldrich, and M. B. Mineev-Weinstein. ROAMing terrain: Real-time optimally adapting meshes. In *Proc. of the Conference on Visualization (VIS)*, pages 81–88, 1997.

[16] D. S. Ebert, R. M. Rohrer, , C. D. Shaw, P. Panda, J. M. Kukla, and D. A. Roberts. Procedural shape generation for multi-dimensional data visualization. *Computers and Graphics*, pages 375–384, 2000.

[17] N. Elmqvist, T. Do, H. Goodell, N. Henry, and J. D. Fekete. Zame: Interactive large-scale graph visualization. In *Proc. of the IEEE Pacific Visualization Symposium (PacificVIS)*, pages 215 –222, 2008.

[18] M. R. Endsley. Toward a theory of situation awareness in dynamic systems: Situation awareness. *Human Factors*, 37(1):32–64, 1995.

[19] M. R. Endsley, B. Bolté, and D. G. Jones. *Designing for Situtation Swareness: An Approach to User-centered Design*. Taylor & Francis Inc., New York, NY, 2003.

[20] M. R. Endsley and D. J. Garland. *Situation Awareness Analysis and Measurement*. Lawrence Erlbaum Associates, Hillsdale, NJ, 2001.

[21] M. R. Endsley, R. Sollenberger, and E. Stein. Situation awareness: A comparison of measures. In *Proc. of the Human Performance, Situation Awareness and Automation (HPSAA)*, 2000.

[22] K. Engel, M. Hadwiger, J. M. Kniss, A. E. Lefohn, C. R. Salama, and D. Weiskopf. Real-time volume graphics. In *Proc. of the ACM Conference on Computer Graphics and Interactive Techniques (SIGGRAPH)*, 2004.

[23] W. Evans, D. Kirkpatrick, and G. Townsend. Right triangular irregular networks. Technical report, University of Arizona, Tucson, AZ, 1997.

[24] E. Feibush, N. Gagvani, and D. Williams. Visualization for situational awareness. *IEEE Computer Graphics and Applications*, pages 38–45, 2000.

[25] M. Florek and M. Novotny. Interactive information visualization using graphics hardware. In *Poster Proc. of the Spring Conference on Computer Graphics (SCCG)*, 2006.

[26] A. Fredrikson, C. North, C. Plaisant, and B. Shneiderman. Temporal, geographical and categorical aggregations viewed through coordinated displays: A case study with highway incident data. In *Proc. of the Workshop on New Paradigms in Information Visualization and Manipulation (NPIVM)*, pages 26–34, 1999.

[27] M. J. Harris, W. V. Baxter, T. Scheuermann, and A. Lastra. Simulation of cloud dynamics on graphics hardware. In *Proc. of the ACM SIGGRAPH/EUROGRAPHICS Conference on Graphics Hardware*, pages 92–101, 2003.

[28] M. Harrower and C. A. Brewer. Colorbrewer.org: An online tool for selecting colour schemes for maps. *The Cartographic Journal*, 40(1):27–37, 2003.

[29] J. Heer and M. Agrawala. Design considerations for collaborative visual analytics. In *Proc. of the IEEE Symposium on Visual Analytics Science and Technology (VAST)*, pages 171–178, 2007.

[30] J. Heer, S. K. Card, and J. A. Landay. Prefuse: A toolkit for interactive information visualization. In *Proc. of the SIGCHI Conference on Human Factors in Computing Systems (CHI)*, pages 421–430, 2005.

[31] H. Hochheiser and B. Shneiderman. Dynamic query tools for time series data sets: Timebox widgets for interactive exploration. *Information Visualization*, 3:1–18, 2004.

[32] C. M. Hoffman, Y. J. Kim, R. P. Winkler, J. D. Walrath, and P. J. Emmerman. Visualization for situation awareness. In *Proc. of the Workshop on New Paradigms in Information Visualization and Manipulation (NPIVM)*, pages 36–40, 1998.

[33] H. Hoppe. Smooth view-dependent level-of-detail control and its application to terrain rendering. In *Proc. of the Conference on Visualization (VIS)*, pages 35–42,

1998.

[34] D. A. Keim, F. Mansmann, J. Schneidewind, and T. Schreck. Monitoring network traffic with radial traffic analyzer. In *IEEE Symposium On Visual Analytics Science And Technology (VAST)*, pages 123–128, 2006.

[35] D. A. Keim, J. Schneidewind, and M. Sips. Circleview: A new approach for visualizing time-related multidimensional data sets. In *Proc. of the Working Conference on Advanced Visual Interfaces (AVI)*, pages 179–182, 2004.

[36] R. M. Kirby, H. Marmanis, and D. H. Laidlaw. Visualizing multivalued data from 2d incompressible flows using concepts from painting. In *Proc. of the Conference on Visualization (VIS)*, pages 333–340, 1999.

[37] Kitware Inc. VTK Visualization Toolkit. http://www.vtk.org, Accessed on November 1, 2011.

[38] H. Lam, D. Russell, D. Tang, and T. Munzner. Session viewer: Visual exploratory analysis of web session logs. In *Proc. of the IEEE Symposium on Visual Analytics Science and Technology (VAST)*, pages 147–154, 2007.

[39] Lawrence Livermore National Laboratory. Visit. https://wci.llnl.gov/codes/visit, Accessed on November 1, 2011.

[40] Leiden University. Knowledgeflow. http://www.liacs.nl/j̃oost/DM/knowledge.htm, Accessed on November 1, 2011.

[41] P. Lindstrom, D. Koller, W. Ribarsky, L. F. Hodges, N. Faust, and G. A. Turner. Real-time continuous level of detail rendering of height fields. In *Proc. of the ACM Conference on Computer Graphics and Interactive Techniques (SIGGRAPH)*, pages 109–118, 1996.

[42] P. Lindstrom and V. Pascucci. Visualization of large terrains made easy. In *Proc. of the Conference on Visualization (VIS)*, pages 363–371, 2001.

[43] Y. Livnat, J. Agutter, S. Moon, and S. Foresti. Visual correlation for situational awareness. In *Proc. of the IEEE Symposium on Information Visualization (INFOVIS)*, pages 95–102, 2005.

[44] D. Luebke and C. Erikson. View-dependent simplification of arbitrary polygonal environments. In *Proc. of the ACM Conference on Computer Graphics and Interactive Techniques (SIGGRAPH)*, pages 199–208, 1997.

[45] D. Luebke, M. Harris, J. Krüger, T. Purcell, N. Govindaraju, I. Buck, C. Woolley, and A. Lefohn. GPGPU: General purpose computation on graphics hardware. In *Proc. of the ACM Conference on Computer Graphics and Interactive Techniques (SIGGRAPH)*, 2004.

[46] D. P. Luebke and B. Hallen. Perceptually-driven simplification for interactive rendering. In *Proc. of the Eurographics Workshop on Rendering Techniques*, pages 223–234, 2001.

[47] F. Mansmann, D. A. Keim, S. C. North, B. Rexroad, and D. Sheleheda. Visual analysis of network traffic for resource planning, interactive monitoring, and interpretation of security threats. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1105–1112, 2007.

[48] W. R. Mark., R. S. Glanville, K. Akeley, and M. J. Kilgard. Cg: A system for programming graphics hardware in a C-like language. *ACM Transactions on Graphics*, 22:896–907, 2003.

[49] B. McDonnel and N. Elmqvist. Towards utilizing GPUs in information

visualization: A model and implementation of image-space operations. *IEEE Transactions on Visualization and Computer Graphics*, 15(6):1105–1112, 2009.

[50] C. Muelder and K. Ma. Rapid graph layout using space filling curves. In *Proc. of the IEEE Information Visualization Conference (INFOVIS)*, 2008.

[51] National Instruments. Labview. http://www.ni.com/labview, Accessed on November 1, 2011.

[52] J. Nickolls, I. Buck, M. Garland, and K. Skadron. Scalable parallel programming with CUDA. *Queue*, 6:40–53, 2008.

[53] M. Oneppo. HLSL shader model 4.0. In *Proc. of the ACM Conference on Computer Graphics and Interactive Techniques (SIGGRAPH)*, pages 112–152, 2007.

[54] D. Overby, J. Keyser, and J. Wall. Interactive visual analysis of location reporting patterns. In *Proc. of the IEEE Symposium on Visual Analytics Science and Technology (VAST)*, pages 223–224, 2009.

[55] D. Overby, J. Keyser, and J. Wall. ProDV: A case study in delivering visual analytics. In *Proc. of the IEEE Symposium on Visual Analytics Science and Technology (VAST)*, pages 247 –248, 2010.

[56] J. D. Owens, D. Luebke, N. Govindaraju, M. Harris, J. Kruger, A. E. Lefohn, and T. J. Purcell. A survey of general-purpose computation on graphics hardware. *Computer Graphics Forum*, 26(1):80–113, 2007.

[57] R. Pajarola. Large scale terrain visualization using the restricted quadtree triangulation. In *Proc. of the Conference on Visualization (VIS)*, pages 19–26, 1998.

[58] C. Plaisant, B. Milash, A. Rose, S. Widoff, and B. Shneiderman. Lifelines: Visualizing personal histories. In *Proc. of the SIGCHI Conference on Human Factors in Computing Systems (CHI)*, pages 221–227, 1996.

[59] F. H. Post, F. J. Post, T. V. Walsum, and D. Silver. Iconic techniques for feature visualization. In *Proc. of the Conference on Visualization (VIS)*, pages 288–295, 1995.

[60] Rapid-I Inc. Rapidminer. http://rapid-i.com/content/view/181/196/, Accessed on November 1, 2011.

[61] W. Ribarsky, E. Ayers, J. Eble, and S. Mukherjea. Glyphmaker: Creating customized visualizations of complex data. *Computer*, 27:57–64, 1994.

[62] R. J. Rost, J. M. Kessenich, and B. Lichtenbelt. *OpenGL Shading Language*. Addison-Wesley, Boston, MA, 2004.

[63] W. Schroeder, K. M. Martin, and W. E. Lorensen. *The Visualization Toolkit: An Object-oriented Approach to 3D Graphics*. Prentice-Hall, Upper Saddle River, NJ, 1998.

[64] B. Shneiderman. The eyes have it: A task by data type taxonomy for information visualizations. In *Proc. of the IEEE Symposium on Visual Languages*, pages 336–343, 1996.

[65] B. Shneiderman and C. Plaisant. Strategies for evaluating information visualization tools: Multi-dimensional in-depth long-term case studies. In *Proc. of Beyond Time and Errors: Novel Evaluation Methods for Information Visualization (BELIV)*, pages 1–7, 2006.

[66] B. Technologies. Openmap: Open systems mapping technology. http://openmap.bbn.com, Accessed on November 1, 2011.

[67] D. R. Tesone and J. R. Goodall. Balancing interactive data management of massive data with situational awareness through smart aggregation. In *Proc. of the IEEE Symposium on Visual Analytics Science and Technology (VAST)*, pages 67 –74, 2007.

[68] University of Konstanz. Knime. http://www.knime.org/knime, Accessed on November 1, 2011.

[69] U.S. Department of Defense. *MIL-STD-2411*. 1994.

[70] U.S. Department of Defense. *MIL-PRF-89020B*. 2000.

[71] T. D. Wang, C. Plaisant, A. J. Quinn, R. Stanchak, S. Murphy, and B. Shneiderman. Aligning temporal data by sentinel events: Discovering patterns in electronic health records. In *Proceeding of the SIGCHI Conference on Human Factors in Computing Systems (CHI)*, pages 457–466, 2008.

[72] C. Weaver, D. Fyfe, A. Robinson, D. Holdsworth, D. Peuquet, and A. M. MacEachren. Visual analysis of historic hotel visitation patterns. In *IEEE Symposium on Visual Analytics Science And Technology (VAST)*, pages 35–42, 2006.

[73] C. M. Wittenbrink, A. T. Pang, and S. K. Lodha. Glyphs for visualizing uncertainty in vector fields. *IEEE Transactions on Visualization and Computer Graphics (TVCG)*, 2:266–279, 1996.

[74] M. Woo, J. Neider, T. Davis, and D. Shreiner. *OpenGL Programming Guide: The Official Guide to Learning OpenGL, Version 1.2*. Addison-Wesley Longman Publishing, Boston, MA, 1999.

[75] Y. Zhang, Q. Huang, and J. Han. Real-time rendering of large-scale terrain based on GPU. In *IEEE Conference on Industrial Electronics and Applications (ICIEA)*, pages 3795–3799, 2009.

APPENDIX A

USER STUDY

This appendix includes the questionaire used during the user evaluation study described in Chapter V.

A. Introduction Script and Data Description

You will first be introduced to general capabilities of the visualization interface. You will be allowed to explore these functions using an interactive bargraph until you are comfortable with the interface. You will then be shown 3 different datasets using 3 different interactive visualization methods. For each visualization method, you will be asked to answer a series of 10 analytic questions about the data shown. The data shown using each visualization consists of one day of sample data for a vehicle location system. The system in question is configured to send location reports for all vehicles at least once every five (5) minutes, or sooner if the vehicle moves.

You may choose to skip a question if you feel that the capabilities provided by the given system are inadequate to answer the question. Otherwise, you are encouraged to answer each question to the best of your ability.

Familiarization using bargraph (on MCS visualization) until user is ready. When ready, ask user to find the value of HTTP traffic at 14:00 (2:00 pm). Answer is 82170.

The visualization environment provides four primary tools, located at the top left of the window. You can choose the Select, Pan, Zoom, or Info tool. Only one tool can be selected at a time. The Select, Pan, and Zoom tools operate on any of the visualizations as expected. The mouse wheel can also be used to zoom in or out while using other tools. The info tool provides additional information on the values shown using each visualization.

Brief information is shown when hovering with the Info tool; more detailed information is shown when the user clicks on a location within a visualization window.

B. Proposed Visualization Method Description

This visualization method will depict the period of time between each location report using rows of isosceles triangles (one for each vehicle), where the left edge of the triangle correspond to the time when the report was sent, and the right tip of the triangle indicates the time the next report was sent. The triangle's color will fade quickly to red if the length of the period exceeds the specified threshold (default is 301 seconds). If the period is over the threshold, the triangle will show some amount of red regardless of the zoom level. Depending on the zoom level, if the current triangle's width is less than 2 pixels, the triangle will merge with other small triangles surrounding it to form a solid rectangle.

Allow user to explore interaction with the visualization on another day (not the day he will be answering questions on) until he/she is ready, before beginning questions. Start timing of each question when participant/moderator finishes reading the question.

C. Simplified Visualization Method Description

This visualization method will depict the period of time between each location report using a vertical line to indicate the time of each location report per vehicle. The line will be red if the length of time between the current location report and the last is over the threshold.

Allow user to explore interaction with the visualization on another day (not the day he will be answering questions on) until he/she is ready, before beginning questions. Start timing of each question when participant/moderator finishes reading the question.

D. Line Chart/State Plot Visualization Description

This visualization method uses a traditional plotting method to show the number of position reports for each vehicle over time. The number of reports have been calculated for 5 minute increments over the course of the day.

Allow user to explore interaction with the visualization on another day (not the day he will be answering questions on) until he/she is ready, before beginning questions. Start timing of each question when participant/moderator finishes reading the question.

E. Questions (to be answered for each of three days)

1. Please count the number of times during the day 4 or more vehicles started moving together (within 5 minutes of each other).
2. Is there a vehicle whose frequency pattern remained constant for the entire time it was reporting?
3. How many vehicles had periods with no reports for longer than 20 minutes?
4. Which went over the 5 minute threshold the most, and how many times?
5. How many vehicles experienced no periods between reports over the 5 minute threshold?
6. Which vehicle experienced the longest period between reports?
7. Which vehicle experienced the longest period of motion?
8. Which vehicle was reporting for the shortest amount of time?
9. Are there any synchronized periods of no reporting (ie 4 or more vehicles stop reporting [go over 5-minute threshold] at the same time)? If so, when?

10. Can you find a vehicle with a prolonged irregular pattern (ie pattern has periods of different length for at least one hour)? If so, which vehicle?

VITA

Name:	Derek Robert Overby
Email:	dereko@tamu.edu
Web:	http://students.cse.tamu.edu/dereko/
Address:	Texas Center for Applied Technology,
	Texas A&M University
	3407 TAMU
	College Station, Texas 77843-3407
Education:	B.S., Computer Science, Texas A&M University, 2000
	M.S., Computer Science, Texas A&M University, 2002
	Ph.D., Computer Science, Texas A&M University, 2011